

(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開2002-237808

(P2002-237808A)

(43)公開日 平成14年8月23日(2002.8.23)

(51)Int.Cl. <sup>7</sup>	識別記号	FI	テ-マ-ト*(参考)
H04L 9/08		G06F 17/60	142 5J104
G06F 17/60	142	H04L 9/00	601A

審査請求 未請求 請求項の数27 OL (全 42 頁)

(21)出願番号 特願2001-33915(P2001-33915)

(22)出願日 平成13年2月9日(2001.2.9)

(71)出願人 000003078

株式会社東芝

東京都港区芝浦一丁目1番1号

(72)発明者 上林 達

神奈川県川崎市幸区小向東芝町1番地 株  
式会社東芝研究開発センター内

(74)代理人 100083161

弁理士 外川 英明

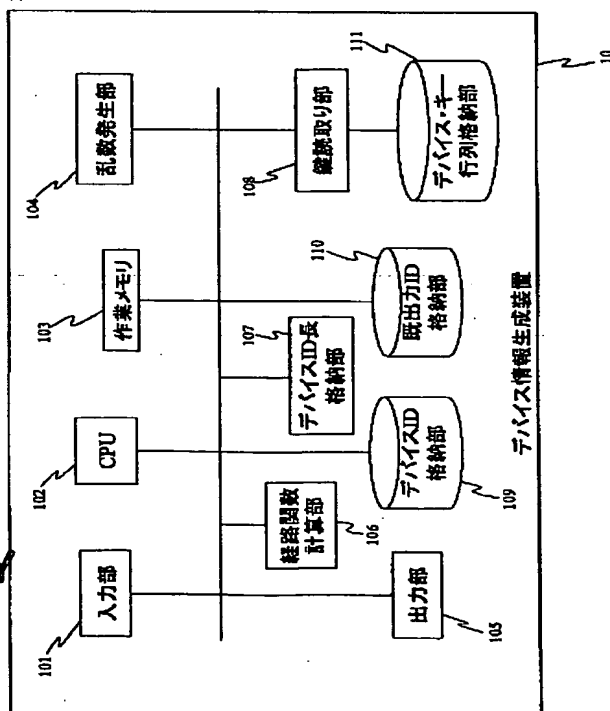
Fターム(参考) 5J104 AA12 AA16 EA06 EA17 NA02

(54)【発明の名称】 デバイス情報生成装置、デバイス情報生成方法、制御データ生成装置、制御データ生成方法、情報利用装置、情報利用方法、および記憶媒体

(57)【要約】

【課題】 利用デバイスにおけるデータの利用を、制御データによって制御する技術において、重大なセキュリティ上の問題と、リボケーションに際して発生し、ユーザーの利便性を大きく損なう副作用の問題とを、共に回避するデバイス情報生成装置を提供する。

【解決手段】 利用デバイス30に割り当てられる複数の数の組からなるデバイス識別子を入力し、このデバイス識別子の部分経路に対応する経路関数値を、計算する経路関数計算部106を備えた。



BEST AVAILABLE COPY

## 【特許請求の範囲】

【請求項1】 情報処理装置に割り当てられる複数の数の組からなるデバイス識別子を入力し、このデバイス識別子の部分経路に対応する経路関数値を、少なくとも一つ生成する生成手段とを具備することを特徴とするデバイス情報生成装置。

【請求項2】 情報処理装置に割り当てられる複数の数の組からなるデバイス識別子を生成する第1生成手段と、このデバイス識別子の部分経路に対応する経路関数値を、少なくとも一つ生成する第2生成手段とを具備することを特徴とするデバイス情報生成装置。

【請求項3】 デバイス情報を生成するデバイス情報生成装置であって、数字の並びであるデバイス識別子を構成する数字の一部または全部の数字の並びである、部分経路の少なくとも一つに関して、当該部分経路における経路関数値を計算する、経路関数計算部を具備する事を特徴とする、デバイス情報生成装置。

【請求項4】 請求項3のデバイス情報生成装置であって、互いに相異なる入力経路に対して互いに相異なる経路関数値を対応させる、経路関数計算部を具備する事を特徴とする、デバイス情報生成装置。

【請求項5】 請求項1のデバイス情報生成装置であって、数字の並びである経路に対応して定まる、デバイス・キー行列の複数の成分に依存する経路関数値を出力する、経路関数計算部を具備する事を特徴とする、デバイス情報生成装置。

【請求項6】 制御データを生成する制御データ生成装置であって、入力された経路に属する数字の一部または全部を用いて作られる部分経路を、少なくとも一つ出力する、随伴頂点集合計算部を具備する事を特徴とする、制御データ生成装置。

【請求項7】 制御データを生成する制御データ生成装置であって、経路を少なくとも一つ設定することができ、入力された経路が、設定された経路に属する数字の一部または全部を用いて作られる部分経路に一致するか否かを判定し、判定結果を出力する随伴頂点集合判定部を具備する事を特徴とする、制御データ生成装置。

【請求項8】 制御データを生成する制御データ生成装置であって、経路を少なくとも一つ設定することができ、入力された経路が、設定された経路に少なくとも一つの数字を追加して作られる経路に一致するか否かを判定し、判定結果を出力する随伴頂点集合判定部を具備する事を特徴とする、制御データ生成装置。

【請求項9】 制御データを生成する制御データ生成装置であって、入力された経路に少なくとも一つの数字を追加して作られる経路を、少なくとも一つ出力する、随伴頂点集合計算部を具備する事を特徴とする、制御データ生成装置。

【請求項10】 制御データを生成する制御データ生成装置であって、入力された経路に属する数字の一部また

は全部を用いて作られる部分経路について、当該部分経路に属する数字の一部または全部を変更して作られる経路を少なくとも一つ出力する、境界集合計算部を具備する事を特徴とする、制御データ生成装置。

【請求項11】 制御データを生成する制御データ生成装置であって、入力された経路に対して定まる経路関数値を計算する、経路関数計算部を具備する事を特徴とする、制御データ生成装置。

【請求項12】 請求項11の制御データ生成装置であって、互いに相異なる入力経路に対して互いに相異なる経路関数値を対応させる、経路関数計算部を具備する事を特徴とする、制御データ生成装置。

【請求項13】 請求項11の制御データ生成装置であって、入力された経路に対して定まる、デバイス・キー行列の複数の成分に依存する経路関数値を出力する、経路関数計算部を具備する事を特徴とする、制御データ生成装置。

【請求項14】 制御データを生成する制御データ生成装置であって、経路と当該経路に対応付けられた数値の組を、少なくとも一組出力する事を特徴とする、制御データ生成装置。

【請求項15】 請求項14の制御データ生成装置であって、入力された経路に属する数字の一部または全部を用いて作られる部分経路について、当該部分経路に属する数字の一部または全部を変更して作られる経路を少なくとも一つ出力する、境界集合計算部を具備し、出力される経路が当該境界集合計算部の出力である事を特徴とする、制御データ生成装置。

【請求項16】 請求項11、12または13のいずれかの制御データ生成装置であって、当該制御データ生成装置は、経路と当該経路に対応付けられた数値の組を少なくとも一組出力し、経路に対応づけられた数値は、当該経路の入力に対する経路関数計算部の出力値を鍵として、データ利用に必要な情報であるマスター鍵を暗号化する事によって得られる、リボケーション関数値である事を特徴とする、制御データ生成装置。

【請求項17】 請求項6乃至14の何れかの制御データ生成装置が出力した制御データを含む事を特徴とする、記録媒体。

【請求項18】 データ利用を行う情報利用装置であって、デバイス識別子と、デバイス識別子の一部または全部の数字を並べて作られる部分経路に対応する数値が、少なくとも一つ予め格納されている、デバイス情報格納部を具備する事を特徴とする、情報利用装置。

【請求項19】 データ利用を行う情報利用装置であって、請求項1乃至4何れかのデバイス情報生成装置が出力したデバイス情報を予め格納するデバイス情報格納部を具備する事を特徴とする、情報利用装置。

【請求項20】 データ利用を行う情報利用装置であって、経路と当該経路に対応付けられた数値の組を少なく

とも一組含む制御データを読みこむ、制御データ入力部を具備する事の特徴とする、情報利用装置。

【請求項 2 1】 データ利用を行う情報利用装置であって、請求項 6 乃至 1 6 の何れかの制御データ生成装置が出力した制御データを読みこむ、制御データ入力部を具備する事の特徴とする、情報利用装置。

【請求項 2 2】 請求項 1 8 乃至 2 1 記載の情報利用装置において、入力された制御データの中から、デバイス識別子の一部または全部の数字を並べて得られる、部分経路に一致する経路を検索する事の特徴とする、情報利用装置。

【請求項 2 3】 請求項 2 2 記載の情報利用装置であって、入力された制御データの中から、デバイス ID の一部または全部の数字を並べて得られる、部分経路に一致する経路を検索し、制御データ中に、そのような経路が見つからなかった場合、当該利用デバイスがリボークされていると判断し、予め定められたリボーク処理を行う事の特徴とする、情報利用装置。

【請求項 2 4】 請求項 2 2 の情報利用装置であって、入力された制御データの中から、デバイス ID の一部または全部の数字を並べて得られる、部分経路に一致する経路を検索し、制御データ中に、そのような経路が見つからなかった場合、制御データから当該経路に対応するリボーク関数値を読み出し、またデバイス情報格納部から当該経路に対応する経路関数値を読み出し、当該経路関数値を鍵としてリボーク関数値を復号する事の特徴とする、情報利用装置。

【請求項 2 5】 情報処理装置に割り当てられる複数の数の組からなるデバイス識別子を入力し、このデバイス識別子の部分経路に対応する経路関数値を、少なくとも一つ生成することを特徴とするデバイス情報生成方法。

【請求項 2 6】 制御データを生成する制御データ生成方法であって、入力された経路に属する数字の一部または全部を用いて作られる部分経路を、少なくとも一つ出力することを特徴とする制御データ生成方法。

【請求項 2 7】 データ利用を行う情報利用装置に、デバイス識別子と、デバイス識別子の一部または全部の数字を並べて作られる部分経路に対応する数値の少なくとも一つを格納することを特徴とする情報利用方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、電子化データの情報利用装置に関わる。また、本発明は、当該利用デバイスにおける電子化データの利用を制御する制御データを作成する、制御データ作成装置に関わる。また、デバイス情報生成装置に関わる。更に、本発明は、当該制御データが記録されたデータ記録媒体に関わる。

【0002】

【従来の技術】利用デバイスにおける電子化データ（以後単にデータと言う）の利用を、制御データによって制

御する技術として、リボケーションと呼ばれる技術が知られている。リボケーションは、しばしば電子化著作物の著作権保護に応用されている。リボケーションの概念を以下に説明する。

【0003】利用デバイスは、予め固有の情報（デバイス情報）を割り当てられており、デバイス内に保持している。デバイス情報は、個々のデバイス毎に異なっている。あるいは、一群のデバイスが同一のデバイス情報を割り当てられていても良い。例えば、A社が製造したCD再生デバイスは全て同一のデバイス情報を持つ、と言うような場合が、それに当たる。

【0004】今、ある利用デバイスにおいて、セキュリティ対策が不十分であった等の理由により、データの不正利用が発生したと仮定する。この場合、不正利用が発生したデバイスにおけるデータ利用を制限／禁止して、被害の拡大を防ぐ事が重要である。

【0005】このため、制御データは、利用デバイスへ配布され、この制御データを利用デバイスで読み取り、当該制御データと、当該利用デバイスが保持するデバイス情報とに基づいて、データ利用の可否を決定する。これにより、少なくとも不正利用が発生した利用デバイスを含む、一群のデバイスはデータの利用が制限／禁止される。制御データは、例えば、放送やネットワークを通じてデータを送信する場合には、データに付随させて利用端末に配布したり、CD-ROMやDVD-ROM等のパッケージ・メディアを通じて流通する場合には、パッケージ・メディアに記録したりして配布される。

【0006】このように、1) 利用デバイスに対して制御データが配布され、2) 利用デバイスは当該制御データを読み込み、3) 利用デバイスは、デバイス情報と制御データに基づいて、データ利用の可否を判断するといった一連の処理により、利用デバイスにおけるデータ利用を制限／禁止する事を、リボケーションと言う。利用デバイスにおけるデータ利用がリボケーションによって制限／禁止されたとき、当該利用デバイスはリボークされたと言う。制御データの配布は、リボケーションを管理する主体（通常は法人）が行う。これをリボーク主体と呼ぶ。

【0007】リボケーションにおいて、データ利用制限／禁止の対象となる利用デバイスを（リボーク）対象デバイスと呼ぶ。リボケーションは、デバイス情報に基づいて行われるため、同一のデバイス情報を持つデバイスは（例え複数存在しても）区別されない。しかし、後述のMK B技術において見られるように、対象デバイスとデバイス情報が異なっている利用デバイスであっても、データ利用が制限／禁止される現象が発生する事がある。この現象を、リボケーションの錯誤と呼ぶ事にする。これはリボケーションの副作用であり、利用デバイスを使用する消費者の便宜を著しく損なう。この種の副作用の発生を極力避ける事は、リボケーション技術にと

って非常に重要である。

【0008】代表的なリボケーション技術の一つとして、リボケーション・リスト(RL)の技術が良く知られている。RLは、制御データであり、通常はデータに付随した形で利用デバイスに配布される。RLには、当該データの利用を禁止するデバイスのデバイス情報が列挙されている。利用デバイスは、データ利用に先立ってRLを読み取り、自分自身のデバイス情報がRLに含まれているか否かをチェックする。デバイス情報がRLに含まれない場合、当該利用デバイスはデータを利用する。一方、デバイス情報がRLに含まれる場合、当該利用デバイスはデータ利用を行わない。RLの改ざんを防ぐ為、RLを暗号化する事がしばしば行われる。

【0009】RL技術では、対象デバイスのデバイス情報を個別に指定する事ができる。また、リボケーションの錯誤は発生しない。ところが、RL技術は、セキュリティ的に看過し難い問題点を有している。リボーク対象デバイスがデータ利用を中止するのは、当該利用デバイスが自分自身のデバイス情報をRL中に見出したが故に、データ利用を言わば「自粛」するからに過ぎない。データ利用に必要な情報の入手は、リボケーションの判断とは独立に行われる。そもそも、リボーク対象デバイスとは、改造等が施された可能性があり、「信用できない」利用デバイスである。リボーク対象デバイスが、データ利用を「自粛」しないよう改造されていれば、RLのリボケーションは全く効果を持たない。

【0010】メディア・キー・ブロック(MKB)の技術は、RL技術が持つ上記のセキュリティ上の問題点を解決する、リボケーション技術の一つである。MKB技術では、まず、デバイス・キー行列KDを用意する。KDの行、列の数をそれぞれ $m, n$ とおく。またKDの $i$ 行 $j$ 列成分を $k[i, j]$ (ただし、 $0 \leq i < n, 0 \leq j < m$ )と表記する。KDの各成分は、例えば、乱数発生器で生成した乱数である。マスター鍵を $K$ とする。マスター鍵は、データ利用に必要な情報の一つである。例えば、データがデータ・キーで暗号化されており、マスター鍵で暗号化されたデータ・キーが、データと共に利用デバイスに配布されている。マスター鍵を得た利用デバイスは、これを用いてデータ・キーを復号し、更にデータを復号して利用する事ができる。

【0011】上記のデバイス・キー行列KDに対して、デバイス情報は、写像 $p: [0, \dots, n-1] \rightarrow [0, \dots, m-1]$ と、 $p$ が誘導するデバイス鍵の列:  $KD(p) = [k[p(0), 0], k[p(1), 1], \dots, k[p(n-1), n-1]]$ の組 $(p, KD(p))$ とである。また、制御データは、 $m$ 行 $n$ 列の行列 $M$ である。 $M$ をMKBと呼ぶ。 $M$ の $i$ 行 $j$ 列成分を $M[i, j]$ と表記する。 $M$ の初期値(即ち、リ

ボケーションがないときの値)は、 $M[i, j] = \text{Enc}(k[i, j], K)$ で与えられる。 $\text{Enc}$ は、適当な暗号化アルゴリズムによる暗号化関数である。鍵 $w$ によってデータ $x$ を暗号化した結果を $\text{Enc}(w, x)$ と表記する。

【0012】デバイス情報 $(p, KD(p))$ を有する利用デバイスは、MKBMを読み取り、図39に示す処理を行なう。この処理をMKB処理と呼ぶ。MKB処理において、 $\text{Dec}$ は、暗号化関数 $\text{Enc}$ に対応する復号関数である。 $\text{Dec}(w, x)$ は、データ $x$ を鍵 $w$ で復号した結果を示す。定義から明らかに、 $\text{Dec}(w, \text{Enc}(w, x)) = x$ となる。

【0013】また、 $\text{null}$ は予約された特別な数値である。特に $\text{null} \neq K$ でなければならない。 $p$ 及び $KD(p)$ は、それぞれ配列 $P$ 及び $KDP$ に格納されているものとする。 $P[j] = p(j)$ 、 $KDP[j] = k[p(j), j]$ と示す。

【0014】 $N\text{Num}$ は多倍長整数のクラスである。 $M$ の初期値に対して、リボーク処理が、デバイス情報 $(p, KD(p))$ に関わりなく $\text{Result} = K$ を与える事は容易に読取れる。

【0015】今、デバイス情報 $(a, KD(a))$ を有する利用デバイス $D$ をリボークする事を考える。この時、リボーク主体は、次のMKB  $M'$ を利用デバイスに配布する。

$M'[a(j), j] = \text{Enc}(k[a(j), j], \text{null})$

$M'[i, j] = \text{Enc}(k[i, j], K)$  if  $i \neq a(j)$

明らかに、利用デバイス $D$ がMKB  $M'$ を処理して得る結果は $\text{null}$ であり、マスター鍵 $K$ を得る事はできない。一方、 $D$ 以外の利用機器 $D'$ が $M'$ を処理すれば、 $K$ を得る。利用機器 $D$ は、しかも $D$ だけ、マスター鍵 $K$ を得る事ができない。その結果、利用機器 $D$ はデータ復号等の処理に進むことができず、リボークされた事になる。

【0016】MKB技術において、MKB処理を実行するのは利用機器である。しかし、RLの場合とは異なり、リボーク処理を行なうか否かの判断は利用機器に依存していない。ある利用機器がメディア・キー・ブロックによってリボークされている場合、当該利用機器に割り当てられているデバイス情報をどのように利用しても、マスター鍵を取り出す事はできない。従って、MKB技術は、RLが持つ先述のセキュリティ上の問題点を解決している。

【0017】MKB技術には、しかし、重大な欠点が存在する。即ち、複数の利用機器がリボークされた場合、リボケーションの錯誤が発生する可能性がある。簡単の為に小さなMKBを用いて、この事を説明する。デバイス・キー行列のサイズを4行4列とし、MKB  $M$ の成分が次の様に与えられているとする。

$M[2, 0] = \text{Enc}(k[2, 0], \text{null})$ ,  $M[2, 1] = \text{Enc}(k[2, 1], \text{null})$ ,

$M[1, 2] = \text{Enc}(k[1, 2], \text{null})$ ,  $M[3, 3] = \text{Enc}(k[3, 3], \text{null})$ ,

$M[i, j] = \text{Enc}(k[i, j], K)$  (上記以外)

上記MKBでは、デバイス情報(  $p$ ,  $KD[p]$  )を持つ利用デバイスD2が(そして、D2だけが)リポークされている。ただし、 $p = [2, 2, 1, 3]$ 。

【0018】ここで更に、デバイス情報(  $p'$ ,  $KD[p']$  )

$$\begin{aligned} M'[2, 0] &= M[2, 0], M'[1, 0] = \text{Enc}(k[1, 0], \text{null}), \\ M'[2, 1] &= M[2, 1], M'[3, 1] = \text{Enc}(k[3, 1], \text{null}), \\ M'[1, 2] &= M[1, 2], \\ M'[3, 3] &= M[3, 3], M'[2, 3] = \text{Enc}(k[2, 3], \text{null}) \\ M'[i, j] &= \text{Enc}(k[i, j], K) \end{aligned}$$

$M'$ によって、利用デバイスD2及びD3は確かにリポークされる。しかし、同時に、例えばデバイス情報(  $p'$ ,  $KD[p']$  )を有する利用デバイスまでリポークされている。ただし、 $p' = [2, 3, 1, 3]$ 。D2、D3以外にはもちろんリポークされてしまう利用デバイスは、全部で6台ある。

【0019】MKB技術においては、一般に、 $s$ 台の利用デバイス( $s$ 個のデバイス情報)をリポークした場合、最大 $sn - s$ 台の利用デバイスが、リポーケーションの錯誤によってリポークされる。「無実の罪」を着せられた利用デバイスのユーザーも、リポーク対象デバイスのユーザーと共に、データ利用を制限される不都合を被る事となる。場合によっては、この事が重大な経済的損失等に繋がる可能性も否定できない。MKBをリポーケーション技術として採用した場合、利用デバイスの供給業者や製造業者は、ユーザーからの苦情や損害賠償の要求など、潜在的な製品トラブルを抱える事になると言っても過言ではない。

【0020】MKB技術において、ある利用デバイスがリポーケーションの錯誤によってリポークされる確率は、一般的な仮定の下で、リポーク対象デバイスの数に対して指数的に増大する。この事は、実際には、デバイス情報全体の中で極く少数のデバイス情報しかリポークできないと言う事を意味している。利用デバイス供給業者或いは製造業者も、錯誤の数が少数に止まっている間は、苦情に個別に対応しトラブルを処理する事が可能である。

【0021】MKB技術を用いたリポーケーションに必要な装置と、その構成及び動作を、以下若干詳細に述べる。これは、MKB技術と本発明との相違を明確に述べる為である。MKB技術におけるデバイス情報生成装置50の構成を、図40に示す。また、その動作を、図41及び図42に示す。デバイス・キー行列KDは二次元の配列として、デバイス・キー格納部509に格納されている。

$$k[0, 0] \ k[0, 1] \ k[0, 2]$$

$$k[1, 0] \ k[1, 1] \ k[1, 2]$$

デバイス・キー行列自体は、乱数発生器を用いるなど適当な方法により、予め作成されているものとする。乱数発生部504は、乱数発生指示を受け取ると、0または1の値をとる乱数を3個発生する。鍵読取り部508は、

によって指定される利用デバイスD3をリポークする必要が生じたとする。ただし、 $p' = [1, 3, 1, 2]$ 。MKBは更新され、 $M'$ となる。 $M'$ の各成分は次の様に与えられる：

(上記以外)

乱数を受け取り、当該乱数を行番号と見なし、デバイス・キー行列の各列から順に、当該行番号によって指定される要素を読みとる。

【0022】機器情報格納部506には、次の二つの情報が記憶される。

行位置の並び： $[R0, R1, R2]$

鍵の並び： $[k[R0, 0], k[R1, 1], k[R2, 2]]$

これらが、機器情報を構成する。既出力情報格納部507には、行位置の並びが追加書きで記録される。従って、既出力情報格納部507には、出力された行位置の並びが全て記録されている。

【0023】次にMKB生成装置の構成を図43に示す。図44及び図45にMKB生成時の動作を示す。先の機器情報生成装置50の場合と同様、デバイス・キー行列は、先述のデバイス・キー行列生成装置を用いて既に生成されていて、下記の二次元配列として、デバイス・キー行列格納部612に格納されているものとする。

$$K[0, 0] \ K[0, 1] \ K[0, 2]$$

$$K[1, 0] \ K[1, 1] \ K[1, 2]$$

鍵読取り部611は、二つの引数を受け取る。二つの引数は、デバイス・キー行列の行と列の番号であり、鍵読取り部611は、これらの番号によって指定されるデバイス・キー行列の要素を返す。MKBは、二次元の配列として、メディア・キー・ブロック格納部610に格納される。

$$M[0, 0] \ M[0, 1] \ M[0, 2]$$

$$M[1, 0] \ M[1, 1] \ M[1, 2]$$

なお、図44及び図45において、 $i, j$ は配列の添字を格納する為の変数であり、例えば、CPU605内のメモリに割り当てられる。

【0024】図46はMKB更新時のMKB生成装置60の動作である。リポークすべき利用機器の機器情報を指定する必要があるが、それはMKBの各列の行番号で指定される。例えば、リポークすべき利用機器のデバイス情報(  $(1, 0, 0)$ ,  $KD(1, 0, 0)$  )とする。この場合、MKB生成装置60のリポーク情報入力部602に、次のデータを入力して指定する。

$$(1(0), 1(1), 1(2)) = (1, 0, 0).$$

CPU605は、これを順に零列目、一列目、及び二列目の行番号と見なし、それぞれをMKBの配列要素の指定、即ち行番号と列番号のペア、に変換して更新部60

9に順次入力する。更新部609は入力された、当該行番号と列番号のペアによって指定されるメディア・キー・ブロックの要素を無効化する。

【0025】MKBに基づくリボケーション方式に準拠する利用機器70の構成を図47に示す。また、その動作を図48及び図49に示す。利用機器70のMKB入力部701は、次のMKBを読み込む。

$M[0, 0] \ M[0, 1] \ M[0, 2]$

$M[1, 0] \ M[1, 1] \ M[1, 2]$

読み込まれたMKBは、MKB格納部702に格納される。デバイス情報格納部705は、例えば、機器情報(1, KD(1))を格納している。なお、KD(1)の各成分を、 $KD(1) = (k(0), k(1), k(2))$ のように表記する。

【0026】CPU703は、デバイス情報格納部705からデータを順に読取り、MKBに適用して行く。即ち、先ず変数jの値を1として、デバイス情報格納部705から $1(j)$ を読取り、配列要素読取り部707に数の組 $(1(j), j)$ を送る。配列要素読取り部707は、MKB格納部702から要素 $M[1(j), j]$ を読み出してCPU703に返す。CPU703は、 $M[1(j), j]$ を復号部708に送る。

【0027】次いで、機器情報から $k(j)$ を読取り、それを復号部708に送る。復号部708は、鍵 $k(j)$ で $M[1(j), j]$ を復号し、その結果をCPU703に返す。CPU703は、当該復号結果を、変数Resultに一時的に格納する。変数Resultの値がnullに等しければ、CPU703はjを1増やす。j<3なら、CPU703は上記動作を繰り返す。さもなければ、CPU703はメディア・キー・ブロック処理動作を停止する。

【0028】j=0, 1, 2のいずれかに対して、変数Resultの値がnullと異なっていた場合、CPUはResultの値を、マスター鍵として、データ利用部709に送る。次いで、CPU703はデータ入力部706からデータを読取り、当該データをデータ利用部709に送る。データ利用部709は、例えば、当該マスター鍵を用いて当該データを復号し、復号結果を利用する。利用部709には、データを利用する為の適当なアルゴリズムが予め格納されているものとする。

#### 【0029】

【発明が解決しようとする課題】本発明は、利用デバイスにおけるデータの利用を、制御データによって制御する技術であって、従来技術に一般的に見られる二つの問題点を解決するものである。即ち、

##### 1. セキュリティ上の問題

データ利用が利用デバイスにおける判断にのみ依存する方式は、リボケーションの実効性そのものに疑義がある。

##### 2. リボケーションの錯誤の問題

リボケーションの錯誤は一種の「冤罪」であり、一般の

善良な利用者の便宜を著しく損ねる。利用デバイス供給・販売業者にとっては、製品を巡るトラブルの要因になる可能性があり、看過できない。本発明は、上記二つの問題点を同時に解消することを目的とする。

#### 【0030】

【課題を解決するための手段】上記課題を解決するために、情報処理装置に割り当てられる複数の数の組からなるデバイス識別子を入力し、このデバイス識別子の部分経路に対応する経路関数値を、少なくとも一つ生成するようにした。

【0031】また、制御データを生成する制御データ生成方法であって、入力された経路に属する数字の一部または全部を用いて作られる部分経路を、少なくとも一つ出力するようにした。

【0032】また、データ利用を行う情報利用装置に、デバイス識別子と、デバイス識別子の一部または全部の数字を並べて作られる部分経路に対応する数値の少なくとも一つを格納するようにした。

【0033】これらによって、利用デバイスにおけるデータの利用を、制御データによって制御する技術において、重大なセキュリティ上の問題と、リボケーションに際して発生し、ユーザーの利便性を大きく損なう副作用の問題とを、共に回避する事が可能となった。

#### 【0034】

【発明の実施の形態】以下、図面を用いて本発明の実施の形態を説明する。

【0035】本発明の詳細な説明の前に、まずは本発明の概要を具体例を挙げて直感的に説明する。

【0036】各利用デバイスは、4個の数字からなるIDを与えられているものとする。各数字は0、1または2の値をとる。従って、今の場合、IDの総数は $3^4=81$ 個である。リボーク主体は、3行4列の行列Kを用意する。Kの成分は、例えば、乱数発生器などで生成した非負整数の乱数とする。Kのi行j列成分を $k(i, j)$ と表記する。Kをデバイス・キー行列と呼び、Kの成分をデバイス・キー(デバイス鍵)と呼ぶ。

【0037】リボーク主体は、各デバイスに対して、デバイス鍵の列を次のように割り当てる：

$(\rho(1), \rho(2), \rho(3), \rho(4))$

と言うデバイスのIDを持つデバイスに対して、デバイス鍵の列

$[k(\rho(1), 1), k(\rho(2), 2), k(\rho(3), 3), k(\rho(4), 4)]$

を割り当てる。

【0038】例えば、(0, 2, 0, 1)と言うデバイスIDを割り当てたデバイスに対して、次のデバイス鍵の列を割り当てる：

$[k(0, 1), k(2, 2), k(0, 3), k(1, 4)]$ 。

各デバイスは、IDと共に、割り当てられたデバイス鍵の列を保持する。

【0039】さて、リボーク対象デバイスが存在しない初期状態で、リボーク主体は次のような制御データを配

$$[(0, \text{Enc}(k(0, 1), K)), (1, \text{Enc}(k(1, 1), K)), (2, \text{Enc}(k(2, 1), K))] \dots \text{式1}$$

【0040】ここに、 $\text{Enc}()$ は適当なアルゴリズムによる暗号化を示す関数である： $\text{Enc}(w, X)$ は、鍵 $w$ によって、平文データ $X$ を暗号化した結果を表す。ここでは、暗号化鍵、平文及び暗号文は非負整数と見なす。なお、利用デバイスは、 $\text{Enc}()$ に対応する復号関数 $\text{Dec}()$ を具備しているものとする。 $\text{Dec}(w, \text{Enc}(w, X)) = X$ 、が任意の鍵 $w$ と任意のデータ $X$ に対して成立する。

【0041】 $K$ はマスター鍵である。 $K$ は利用デバイスが

$$[(0), a(0)], [(2), a(2)], [(1, 0), a(1, 0)], [(1, 1), a(1, 1)], [(1, 2, 1), a(1, 2, 1)], [(1, 2, 2), a(1, 2, 2)], [(1, 2, 0, 0), a(1, 2, 0, 0)], [(1, 2, 0, 2), a(1, 2, 0, 2)] \dots \text{式2}$$

ここで、 $a(x_1) = \text{Enc}(PF(x_1), K)$ ,  $a(x_1, x_2) = \text{Enc}(PF(x_1, x_2), K)$ ,  $a(x_1, x_2, x_3) = \text{Enc}(PF(x_1, x_2, x_3), K)$ ,  $a(x_1, x_2, x_3, x_4) = \text{Enc}(PF(x_1, x_2, x_3, x_4), K)$ とする。

ここで、 $PF(x_1) = k(x_1, 1)$ ,  $PF(x_1, x_2) = k(x_1, 1) \odot k(x_2, 2)$ ,  $PF(x_1, x_2, x_3) = k(x_1, 1) \odot k(x_2, 2) \odot k(x_3, 3)$ ,  $PF(x_1, x_2, x_3, x_4) = k(x_1, 1) \odot k(x_2, 2) \odot k(x_3, 3) \odot k(x_4, 4) \dots \text{式3}$

ただし、非負整数 $x, y$ に対して $x \odot y$ は $x$ と $y$ を二進表現した場合の、排他的論理和を表すものとする。上記 $PF$ を経路関数と呼ぶ。

【0044】上記制御データの作り方は、次の通りである。 $(1, 2, 0, 1)$ と言う数字の並びの先頭から、数字を1個、2個、3個、4個取ったものを並べる： $(1)$ ,  $(1, 2)$ ,  $(1, 2, 0)$ ,  $(1, 2, 0, 1)$ 。この並びを、デ

$(0), (2), (1, 0), (1, 1), (1, 2, 1), (1, 2, 2), (1, 2, 0, 0), (1, 2, 0, 2) \dots \text{式4}$

この数の並びの各要素を変数とする関数 $a$ の値を並べたものが、上記式2である。上のデータの作り方は、木構造を考えると理解し易い。

【0046】図1において、リボーク対象のデバイスID  $(1, 2, 0, 1)$ は点線の経路に対応する。また、式4の経路を実線で示す。それらは点線を辿って、最後だけ実線を辿る。図1の点線と実線と、その始/終点は、所謂三本木をなしている。この木をデバイスID  $(1, 2, 0, 1)$ のリボケーション木と呼ぶ。

【0047】ここで、重要なのは、4つの列全てを通る経路は、点線の経路を除いて、必ずどこかで実線を通らなければならない、と言う点である。式4の経路の集合を、デバイスID  $(1, 2, 0, 1)$ の境界集合と呼ぶ。デバイスID  $(1, 2, 0, 1)$ のリボケーション木の頂点集合は、当該デバイスIDに随伴する頂点集合と、当該デバイスIDの境界集合の和集合である。式2の制御データは、デバイスID  $(1, 2, 0, 1)$ の境界集合に属する各経路に対して、非負整数値を対応させるものである。こ

布する：

データを利用する為に必要な情報である。式1の制御データを初期制御データと呼ぶ。初期制御データは、三つの非負整数からなる。

【0042】今、 $(1, 2, 0, 1)$ というデバイスIDを持つデバイスをリボークする事を考える。このとき、リボーク主体は、次の制御データを作り、利用デバイスに配布する：

$1, x_2, x_3, x_4, K)$ とする。

【0043】

デバイスIDに随伴する頂点集合と呼ぶ。

【0045】次に、上記並びの各要素について、最後の数字を  $(0, 1)$ または  $2$ のいずれかで別の数字に変えたものを列挙する。例えば、最後の数字が  $2$ であった場合、最後の数字を  $0, 1$ に変えた数を列挙する。この操作によって、次の数の並びが得られる。

の対応をリボケーション関数と呼ぶ。この例では、 $a$ がリボケーション関数である。

【0048】式2の制御データを読取った利用デバイスの処理を述べる。利用デバイスのデバイスIDが  $(1, 2, 1, 1)$ であったとする。まず、利用デバイスは、デバイスIDの先頭から一つの数をとって得られる経路  $(1)$ が制御データに含まれるか否かを調べる。即ち、経路  $(1)$ が、デバイスID  $(1, 2, 0, 1)$ の境界集合に属するか否かを調べる。当該制御データは、経路  $(0)$ 及び  $(2)$ （とそれぞれの経路に対するリボケーション関数値）を含むが、 $(1)$ を含まない。この場合、利用デバイスは、デバイスIDの先頭から二つの数をとって得られる経路  $(1, 2)$ について、この経路が制御データに含まれるか否かを調べる。この様に、利用デバイスは、自分自身のデバイスIDに沿って、図1のリボケーション木を順に辿って行く。

【0049】利用デバイスが、 $(1, 2)$ から  $(1, 2, 1)$ まで辿るときに、初めてリボケーション木の実線の枝

を通過する。即ち、経路(1, 2, 1)はデバイスID(1, 2, 0, 1)の境界集合に属しており、従って、当該経路と当該経路に対応するリボケーション関数値が制御データに含まれている。利用デバイスは、当該リボケーション関数値 $a(1, 2, 1)$ を読み取り、次の値を計算する。

$\text{Dec}(\text{PF}(1, 2, 1), a(1, 2, 1))$ ...式5

利用デバイスは、予め割り当てられたデバイス鍵の列 $[k(1, 1), k(2, 2), k(1, 3), k(1, 4)]$ を保持しているので、 $\text{PF}(1, 2, 1)$ を式3に従って計算する事ができる。従って、利用デバイスは、確かに式5の値を計算する事ができる。式5の値は、 $\text{Dec}(\text{PF}(1, 2, 1), \text{Enc}(\text{PF}(1, 2, 1), K)) = K$ に等しい。

【0050】従って、当該利用デバイスはマスター鍵を得て、データを利用する事ができる。即ち、リボークされていない。

【0051】一方、利用デバイスがデバイスID(1, 2, 0, 1)を持っていたとする。上記と同じ手続きにより、利用デバイスは、自分自身のデバイスIDに沿って、図1のリボケーション木を順に辿って行く。今の場合は、しかし、リボケーション木の実線を通る事はない。経路(1), (1, 2), (1, 2, 0), (1, 2, 0, 1)は全て点線しか通らない。従って、どの経路も制御データ中に、対応するリボケーション関数値を見出すことができない。このデバイスは結局マスター鍵Kを得る事無く処理を終える。即ち、このデバイスはリボークされる。

【0052】次に、デバイスID(1, 2, 0, 1)に加えて、デバイスID(1, 1, 1, 0)を持つデバイスをリボークする場合を考える。(1, 2, 0, 1)に随伴する頂点集合と(1, 1, 1, 0)に随伴する頂点集合の和集合を、(1, 2, 0, 1)及び(1, 1, 1, 0)に随伴する頂点集合と言う。

【0053】また、(1, 2, 0, 1)のリボケーション木の頂点集合と(1, 1, 1, 0)のリボケーション木の頂点集合の和集合を、(1, 2, 0, 1)及び(1, 1, 1, 0)のリボケーション木の頂点集合とする。(1, 2, 0, 1)及び(1, 1, 1, 0)のリボケーション木の頂点集合から、(1, 2, 0, 1)及び(1, 1, 1, 0)に随伴する頂点集合を除いたものを、(1, 2, 0, 1)及び(1, 1, 1, 0)の境界集合として定義する。

【0054】具体的には、(1, 1, 1, 0)に随伴する頂点集合は、(1), (1, 1), (1, 1, 1), (1, 1, 1, 0)であるから、(1, 2, 0, 1)及び(1, 1, 1, 0)に随伴する頂点集合は、次のようになる。

(1), (1, 1), (1, 2), (1, 1, 1), (1, 2, 0), (1, 1, 1, 0), (1, 2, 0, 1)。

また、(1, 1, 1, 0)の境界集合は、(0), (2), (1, 0), (1, 2), (1, 1, 0), (1, 1, 2), (1, 1, 1, 1), (1, 1, 1, 2)であるから、(1, 2, 0, 1)及び(1, 1, 1, 0)のリボケーション木の頂点集合は、下記のようになる。

(0), (1), (2), (1, 0), (1, 1), (1, 2), (1, 1, 0), (1, 1, 1), (1, 1, 2), (1, 2, 0), (1, 2, 1), (1, 2, 2), (1, 1, 1, 0), (1, 1, 1, 1), (1, 1, 1, 2), (1, 2, 0, 0), (1, 2, 0, 1), (1, 2, 0, 2)。

従って、(1, 2, 0, 1)及び(1, 1, 1, 0)の境界集合は、下記の通りである。

(0), (2), (1, 0), (1, 1), (1, 1, 0), (1, 1, 2), (1, 2, 1), (1, 2, 2), (1, 1, 1, 1), (1, 1, 1, 2), (1, 2, 0, 0), (1, 2, 0, 2)。

図2に二つのデバイスID(1, 2, 0, 1)及び(1, 1, 1, 0)に随伴する頂点集合と境界集合とを図示する。頂点集合は点線の辺を通る経路の全体に一致する。また、境界集合は点線の辺を辿って最後が実線を通る経路の全体に一致する。

【0055】利用デバイスは、リボーク対象のデバイスIDが一つの場合と同じ手続きで、自分自身のデバイスIDに沿って、図1のリボケーション木を順に辿って行く。

【0056】利用デバイスのデバイスIDが(1, 2, 1, 1)であったとする。利用デバイスの処理が経路(1, 2)から経路(1, 2, 1)に移動する際に、実線を通る。即ち、境界集合に含まれる経路の中で、利用デバイスが初めて出会うものが(1, 2, 1)である。先ほどと全く同様に、制御データに含まれる、対応するリボケーション関数値に対する復号処理により、利用デバイスは、マスター鍵Kを得る事ができる。即ち、当該利用デバイスはリボークされていない。

【0057】一方、利用デバイスのデバイスIDが(1, 1, 1, 0)であったとする。この利用デバイスによる制御データの処理は、デバイスID(1, 1, 1, 0)に随伴する頂点集合だけを通る。それは決して、(1, 2, 0, 1)及び(1, 1, 1, 0)の境界集合に含まれる経路を通らない。当該利用デバイスはリボケーション関数の値を得る事ができず、従って、マスター鍵を得ることはできない。即ち、当該利用デバイスはリボークされる。

【0058】リボーク対象デバイスが三つ以上の場合についても、全く同様である。デバイスIDのリボケーション木を利用する方法によれば、錯誤の副作用を惹き起す事無く、リボーク対象デバイスを個別にリボークする事ができる。しかも、制御データの大きさ(要素数)は、(リボーク対象デバイスの数) × (デバイスIDの長さ) × 2 を超えない。

【0059】なお、デバイスIDをグループでリボークしたい場合も考えられる。例えば、或る利用デバイス・メーカーに(1, 2, \*, \*)と言うデバイスIDを割り当てたとする。ここに\*はワイルド・カード(数字0, 1または2のいずれか)である。この場合、(1, 2, \*, \*)に随伴する頂点集合は、次の様になる。

(1), (1, 2), (1, 2, \*), (1, 2, \*, \*)  
(1, 2, \*, \*)の境界集合は、下記の様になる。



(0), (2), (1, 0), (1, 1).

図3に、(1, 2, \*, \*)に随伴する頂点集合と境界集合とを図示する。

【0060】制御データは、従って、次の様に与えられる。

((0), a(0)), ((2), a(2)), ((1, 0), a(1, 0)), ((1, 1), a(1, 1)).

この制御データによって、デバイスID(1, 2, \*, \*)を持つデバイスが、そして、それらだけがリボークされる。リボークの対象となるデバイスのグループが二つ以上ある場合も、制御データを個別デバイスのリボークと同様に定義する事により、錯誤の無いリボーションが実現される。

[((0), a(0)), ((2), a(2)), ((1, 0), a(1, 0)), ((1, 1), a(1, 1)), ((1, 2, 1), a(1, 2, 1)), ((1, 2, 2), a(1, 2, 2)), ((1, 2, 0, 0), a(1, 2, 0, 0)), ((1, 2, 0, 2), a(1, 2, 0, 2)), ((1), b(1)), ((1, 2), b(1, 2)), ((1, 2, 0), b(1, 2, 0)), ((1, 2, 0, 1), b(1, 2, 0, 1))]

…式6

ここで、

$b(x_1) = \text{Enc}(\text{PF}(x_1), \text{null})$ ,

$b(x_1, x_2) = \text{Enc}(\text{PF}(x_1, x_2), \text{null})$ ,

$b(x_1, x_2, x_3) = \text{Enc}(\text{PF}(x_1, x_2, x_3), \text{null})$ ,

$b(x_1, x_2, x_3, x_4) = \text{Enc}(\text{PF}(x_1, x_2, x_3, x_4), \text{null})$

とする。nullは、マスター鍵が得られない事を示す、予め定められている数値である。今の場合、図1で点線の経路にもリボーション関数値が割り当てられており、その値を定める関数がbである。

【0063】利用デバイスの処理は、先の例と殆ど同様であるが、制御データのチェック方法だけが異なっている。利用デバイスのデバイスIDが(1, 2, 1, 1)であったとする。まず、利用デバイスは、デバイスIDの先頭から一つの数をとって得られる経路(1)に対するリボーション関数値b(1)を制御データから読み取り、次の値を計算する。

$\text{Dec}(\text{PF}(1), b(1))$ 。

この値は、下記に等しい。

$\text{Dec}(\text{PF}(1), \text{Enc}(k_1(1), \text{null})) = \text{null}$ 。

従って、この場合、利用デバイスは、デバイスIDの先頭から二つの数をとって得られる経路(1, 2)について、同じ処理を繰り返す。利用デバイスが、(1, 2, 1)まで辿ったときに、初めてリボーション木の実線を通過する。この場合の処理も同様である。ただし、今回はデバイス鍵Kを得る。

【0064】一方、利用デバイスがデバイスID(1, 2, 0, 1)を持っていたとする。上記と同じ手続きにより、利用デバイスは、自分自身のデバイスIDに沿って、図1のリボーション木を順に辿って行く。今の場合、しかし、リボーション木の実線を通る事はない。

【0061】なお、以上では、制御データは、リボーク対象経路(群)が定める境界集合と、当該集合上のリボーション関数値からなる例を考えた。リボーション関数の定義域をリボーク対象経路(群)に随伴する頂点集合に拡張し、制御データを、リボーション木の頂点集合と、当該頂点集合上のリボーション関数値の組として定義する事もできる。この場合、上述の例より効率は落ちるが、ほぼ同様の効果が得られる。上述の例に則して、これを述べる。

【0062】先の例と同様、(1, 2, 0, 1)というデバイスIDを持つデバイスをリボークする事を考える。このとき、リボーク主体は、次の制御データを作り、利用デバイスに配布する。

経路(1), (1, 2), (1, 2, 0), (1, 2, 0, 1)は全て点線しか通らない。従って、どの経路についても、リボーション関数値の復号によって、nullを得るだけである。このデバイスは結局マスター鍵Kを得る事無く処理を終える。即ち、このデバイスはリボークされる。

【0065】以下に、本発明の第1の実施の形態を詳細に説明する。

【0066】図4は、デバイス情報生成装置10を示したものである。

【0067】リボーク主体は、デバイス情報生成装置10を用いて、デバイスにデバイスIDとデバイス鍵列とを割り当てる。各デバイスにデバイス鍵列を与えることによって、各デバイスは経路関数 $k_1, k_2, k_3$ を計算することが可能である。本実施の形態のデバイス情報生成装置10は、デバイス鍵列ではなく、各デバイス経路関数値を直接与えるようにしたものである。こうすると、各デバイスはリボーク処理の際に、経路関数値計算の手間を省く事ができる。

【0068】デバイス情報生成装置10の動作の一例を図6、図7及び図8を用いて説明する。なお、デバイスIDは、各数字が0, 1または2の値を取りうる3個の数字からなる長さ4であるとする。

【0069】まず、入力部101がデバイス情報生成要求を受け取り、当該要求をCPU102へ送る(S1201)。CPU102が乱数発生部104に、乱数発生を指示すると(S1202)、乱数発生部104によりR1, R2, R3)を発生し、それらをCPU102へ送る(S1203)。CPU102は、送られたR1, R2, R3を作業メモリ103に格納する(S1204)。

【0070】次に、CPUは数の組(R1, R2, R3)を既出力ID格納部110から検索する(S1205)。検

素により、( R1, R2, R3 )が見つかった否かを判定し ( S1206 )、見つかった場合には、ステップ S1203に戻り、再度乱数の発生を行う。一方、( R1, R2, R3 )が見つからなかった場合には、( R1, R2, R3 )を既出力ID格納部110に格納する ( S1207 )。また、( R1, R2, R3 )をデバイスID格納部109へも格納する ( S1208 )。

【0071】CPU102は、R1を経路関数計算部106に送り ( S1209 )、経路関数計算部106にてPF ( R1 )値を計算する ( S1210 )。CPU102は、計算されたPF ( R1 )値をデバイスID格納部109へ格納する ( S1211 )。

【0072】次に、CPU102は、( R1, R2 )を経路関数計算部106に送り ( S1212 )、経路関数計算部106にてPF ( R1, R2 )値を計算する ( S1213 )。CPU102は、計算されたPF ( R1, R2 )値をデバイスID格納部109へ格納する ( S1214 )。

【0073】同様に、CPU102は、( R1, R2, R3 )を経路関数計算部106に送り ( S1215 )、経路関数計算部106にてPF ( R1, R2, R3 )値を計算する ( S1216 )。CPU102は、計算されたPF ( R1, R2, R3 )値をデバイスID格納部109へ格納する ( S1217 )。

【0074】次に、CPU102は、デバイスID格納部109からデバイスID ( R1, R2, R3 )を読み出して、出力部105へ送る ( S1218 )。また、デバイスID格納部109から経路関数値PF ( R1 )、PF ( R1, R2 )、PF ( R1, R2, R3 )を読み出し、出力部105へ送る ( S1219 )。これにより、出力部105は、デバイスID ( R1, R2, R3 )及び経路関数値PF ( R1 )、PF ( R1, R2 )、PF ( R1, R2, R3 )を出力する ( S1220 )。

【0075】経路 ( R1 )、( R1, R2 )及び ( R1, R2, R3 )をデバイスID ( R1, R2, R3 )の部分経路と呼ぶことがある。本実施の形態では、各部分経路に対する経路関数値は次のようにとられる。

$$\begin{aligned} PF(R1) &= \text{Enc}(k(R1, 1), 1) \\ PF(R1, R2) &= \text{Enc}(k(R2, 2), k1(R1)), \\ PF(R1, R2, R3) &= \text{Enc}(k(R3, 3), k2(R1, R2)) \end{aligned}$$

ここで、

$$\begin{aligned} &k(0, 1) \ k(0, 2) \ k(0, 3) \\ &k(1, 1) \ k(1, 2) \ k(1, 3) \\ &k(2, 1) \ k(2, 2) \ k(2, 3) \end{aligned}$$

は、デバイス・キー行列であり、予め乱数発生器などによって生成され、デバイス・キー行列格納部111に格納されている。

【0076】上記したデバイス制御生成装置10の動作の一例において、経路関数値を計算すべき経路pは、( p(1) )、( p(1), p(2) )または、( p(1), p(2), p(3) )と言う形で、経路関数計算部106に入力される。なお、各p(j) ( j=1, 2, 3 )は、数字0, 1

または2である。経路関数計算部106は、入力された経路の長さに応じて、PF ( p(1) )、PF ( p(1), p(2) )またはPF ( p(1), p(2), p(3) )の値を計算し、出力する。この経路関数計算部106の動作の一例を図5を用いて説明する。

【0077】まず、経路pを受け取る ( S1101 )。pの長さを変数1に、V=1、J=1にそれぞれ初期設定する ( S1102～S1104 )。次にJが1より大きいとか否かを判定する ( S1105 )。この場合、J=1なので、否と判定され、数の組(p(j), j)を鍵読み取り部108に送る ( S1106 )。次に、k(p(j), j)を鍵読み取り部108から受け取り ( S1107 )、これを鍵としてVを暗号化し、その結果をVに代入する ( S1108 )。次にJを1だけ増加させ ( S1109 )、ステップS1105の判定に戻る。

【0078】もし、ステップS1105によって、Jが1より大きいと判定されるとVの値を出力し、( S1110 ) 経路関数計算部106の動作は終了する。

【0079】デバイス情報生成装置10は、デバイス情報生成要求を受け取り、デバイスID: ( R1, R2, R3 )、経路関数値: ( PF1, PF2, PF3 )を出力する。これらは、それぞれ三つの符号無し整数の組である。経路関数値は、その作り方から明らかな様に、デバイスIDに依存して定まる。

$$\begin{aligned} PF1 &= PF(R1), \\ PF2 &= PF(R1, R2), \\ PF3 &= PF(R1, R2, R3). \end{aligned}$$

【0080】本実施形態のデバイス情報生成装置10は、MKBのデバイス情報生成装置10と対比されるべきものである。MKBには、経路関数という概念が存在しない。即ち、MKBでは、デバイスIDが通過するMKBの成分だけで、当該デバイスIDがリポーカされるかが決まる。一方、本発明は経路と見なしたデバイスIDをリポーカの対象とするので、デバイスIDが通過する数字を同定する為に、経路関数の値を利用する。本実施形態のデバイス情報生成装置10は、以下の二点に特徴がある。

A) 経路関数計算部106を具備している。経路関数計算部106は、数字の並びであるデバイスIDについて、デバイスIDを構成する数字の一部または全部を用いて作られる部分経路に対する経路関数値を計算する。経路関数値は、経路の長さが1の場合を除き、経路に対応するデバイス・キー行列の複数の成分に依存して定められる数値である。

B) デバイスIDと共に、当該デバイスIDの部分経路に対応する経路関数値を出力する。

【0081】次に、制御データ生成装置20について説明する。図9は制御データ生成装置20の構成を示した図である。

【0082】デバイス・キー行列格納部216には、デバイス・キー行列が格納されている。これは、デバイス情報生成装置10で使用するデバイス・キー行列と同一のものである。

【0083】まず、リボーク対象デバイスが存在しない初期状態における、制御データ生成装置20の動作の一例を図10に示す。

【0084】始めに、マスター鍵 $k$ をマスター鍵入力部201により入力する(S1301)。CPU205は、入力された $K$ をマスター鍵格納部209に送る(S1302)。そして、 $J$ を0に初期設定する(S1303)。

【0085】次に $J$ が3より小さいか否かを判定し(S1304)、 $J$ が3以上になるまで以下の処理を繰り返す。

【0086】まず、CPU205は、マスター鍵格納部209からマスター鍵 $K$ を読む(S1305)。また、CPU205は、数の組( $J, 1$ )を制御データ格納部211に格納する(S1306)。この数の組( $J, 1$ )を鍵読み取り部215に送る(S1307)。

【0087】鍵読み取り部215は、デバイス・キー行列格納部216から $k(J, 1)$ を読み、CPU205へ送る(S1308)。CPU205は $k(J, 1)$ と、 $K$ とを暗号化部207に送る(S1309)。

【0088】暗号化部207は、 $K$ を $k(J, 1)$ で暗号化し、得られた結果 $a(J, 1)$ をCPU205へ送る(S1310)。

【0089】CPU205は、経路と数値の組( $J, 1, a(J, 1)$ )を出力部204へ送り(S1311)、出力部204は、これを出力する(S1312)。

【0090】 $J$ を1だけ増加させ(S1313)、ステップS1304へ戻る。ステップS1304では、 $J$ が3以上になると上記の繰り返しを止め、終了する。

【0091】初期状態における、制御データ生成装置20の出力は、 $((0), a(0)), ((1), a(1)), ((2), a(2))$ である。ここに、 $(0), (1), (2)$ はそれぞれ唯一つの数字からなる(長さ1の)経路である。また、 $a(J)$ は、 $\text{Enc}(k(J, 1), K)$ を示している。

【0092】リボーク対象経路は、数の並びの形でリボーク情報入力部202に入力される。 $(2, 0, 1)$ あるいは $(1, 1)$ などがリボーク対象経路 $p$ として、リボーク情報入力部202に入力される。

【0093】次に、リボーク対象経路 $p$ が初めて入力された時の動作を図11、図12及び図13に示す。

【0094】まず、リボーク情報入力部202に経路 $p$ が入力され(S1401)、作業メモリ206に書き込まれる(S1402)。CPU205は、 $p$ を境界集合計算部208へ送り(S1403)、境界集合計算部208が $p$ の境界集合 $V'$ を求める(S1404)。得ら

れた境界集合 $V'$ を受けたCPU205は、 $V'$ を境界集合格納部210に格納する(S1406)。

【0095】次に、CPU205は、 $V'$ の要素数を変数 $N$ に設定し(S1407)、 $J$ を1に設定する(S1408)。

【0096】これ以降、 $J$ が $N$ より大きいかな否かを判定し(S1409)、この判定結果が否であれば、ステップS1410～S1419を実行する。すなわち、

- ・CPU205が、 $V'$ の $J$ 番目の経路 $p[J]$ を経路関数計算部213に送る(S1410)
- ・経路関数計算部213が $p[J]$ における経路関数値 $k(p[J])$ を計算する(S1411)
- ・CPU205が、 $k(p[J])$ を経路関数計算部213から受け取る(S1412)
- ・CPU205が、マスター鍵 $K$ をマスター鍵格納部209から読み取る(S1413)
- ・CPU205が、数値の組( $k(p[J]), K$ )を暗号化部207に送る(S1414)
- ・暗号化部207が $k(p[J])$ を鍵として $K$ を暗号化。その結果 $a(p[J])$ をCPU205に返す(S1415)
- ・CPU205が経路と数値の組( $p[J], a(p[J])$ )を制御データ格納部211に追加(S1416)
- ・CPU205が( $p[J], a(p[J])$ )を出力部204に送る(S1417)
- ・出力部204が( $p[J], a(p[J])$ )を出力(S1418)
- ・ $J$ を1だけ増分する(S1419)

を $J$ が $N$ より大きくなるまで繰り返す。

【0097】ステップ1409において、 $J$ が $N$ より大きくなったと判断された場合、CPU205が $p$ を作業メモリ206から読み出し(S1420)、随伴頂点集合計算部214に送る(S1421)。随伴頂点集合計算部214が $p$ の随伴頂点集合 $V$ を求め(S1422)、これをCPU205が受ける(S1423)。この受け取った随伴頂点集合 $V$ を随伴頂点集合格納部212に格納し(S1424)、処理を終える。

【0098】以上の動作について解説を加える。リボーク対象経路 $p$ を初めて入力する時点で、制御データ生成装置20の境界集合格納部210と随伴頂点集合格納部212には何も格納されていない。同様に、制御データ格納部211の内容も空である。先に述べた様に、初期状態についても制御データ自体は存在する。制御データ生成装置20は、初期状態で、それを出力するが、当該制御データは制御データ格納部211に格納されない。

【0099】リボーク対象経路 $p$ が入力されると、制御データ生成装置20のCPU205は、 $p$ を境界集合計算部208に送る。境界集合計算部208は、 $p$ の境界集合に属する経路を全て求めて、CPU205に返す。 $p$ の境界集合とは、 $p$ の先頭からとった部分経路の集合に

含まれる各経路の最後の数字を、異なった数字に置き換える事で得られる経路の集合である。例えば、 $p = (2, 0, 1)$ に対して、 $p$ の先頭からとった部分経路の集合は、 $(2)$ 、 $(2, 0)$ 、 $(2, 0, 1)$ となるから、境界集合は、 $(0)$ 、 $(1)$ 、 $(2, 1)$ 、 $(2, 2)$ 、 $(2, 0, 0)$ 、 $(2, 0, 2)$ の各経路の集合である。

【0100】CPU205は、境界集合計算部208から受け取った経路を境界集合格納部210に格納する。

【0101】CPU205は、次いで、境界集合に属する各経路に対するリボケーション関数値を求める。リボケーション関数 $a$ は、経路 $(x_1)$ 、 $(x_1, x_2)$ 、 $(x_1, x_2, x_3)$ に対して、それぞれ次の様に定義される。

$a(x_1) = \text{Enc}(PF(x_1), K)$ ,

$a(x_1, x_2) = \text{Enc}(PF(x_1, x_2), K)$ ,

$a(x_1, x_2, x_3) = \text{Enc}(PF(x_1, x_2, x_3), K)$ 。

式7の $a$ は、長さ1の経路に関するリボケーション関数である。ここで、 $K$ はマスター鍵、 $PF$ は経路関数である。各経路における $PF$ の値は経路関数計算部で計算される。本制御データ生成装置20における経路関数計算部213は、図4のデバイス情報生成装置10における経路関数計算部106と同一のものである。制御データ生成装置20は、境界集合に属する各経路と、当該経路における経路関数値の組を全て出力する。これらの組の全体が、入力されたリボーク対象経路に対する制御データとなる。例えば、リボーク対象経路 $p = (2, 0, 1)$ に対する制御データ生成装置20の出力は、次のようになる。  
 $((0), a(0))$ 、 $((1), a(1))$ 、 $((2, 1), a(2, 1))$ 、 $((2, 2), a(2, 2))$ 、  
 $((2, 0, 0), a(2, 0, 0))$ 、 $((2, 0, 2), a(2, 0, 2))$ 。

次に、CPU205は、経路 $p$ を、随伴頂点集合計算部214に送る。随伴頂点集合計算部214は、 $p$ の随伴頂点集合に属する経路を全て求めて、CPUに返す。デバイスID  $p$ の随伴頂点集合とは、 $p$ の先頭から順にとった部分経路の集合である。即ち $(p(1))$ 、 $(p(1), p(2))$ 、 $(p(1), p(2), p(3))$ の随伴頂点集合は、 $(p(1))$ 、 $(p(1), p(2))$ 、 $(p(1), p(2), p(3))$ の三つの経路の集合である。

【0102】長さがデバイスIDの長さ(本例では3)に満たない経路に関して、随伴頂点集合は次のように定義される。 $(p(1))$ の随伴頂点集合は、 $(p(1))$ 及び $(p(1), ?, ?)$ の形で書ける全ての経路からなる。 $(p(1), p(2))$ の随伴頂点集合は、 $(p(1))$ 、 $(p(1), p(2))$ 及び $(p(1), p(2), ?)$ の形で書ける全ての経路からなる。なお、「 $?$ 」は0、1または2の任意の数字を表すワイルド・カードである。

【0103】要するに、デバイスIDの長さに満たない経路の随伴頂点集合は、当該経路の部分経路と、デバイスIDの長さまでワイルド・カード?を補った形で書ける経路の全体である。デバイスID(2, 0, 1)に随伴する頂

点集合は、 $(2)$ 、 $(2, 0)$ 、 $(2, 0, 1)$ の三つの経路からなる。

【0104】また、例えば経路 $(1, 1)$ の随伴頂点集合は、 $(1)$ 、 $(1, 1)$ 、 $(1, 1, 0)$ 、 $(1, 1, 1)$ 、 $(1, 1, 2)$ の5つの経路からなる。

【0105】CPU205は随伴頂点集合計算部214から受け取った経路を、随伴頂点集合格納部212に格納する。

【0106】以上、初期状態の制御データ生成装置20の動作と、初めてリボーク対象経路を入力した際の制御データ生成装置20の動作を述べた。

【0107】次に、二つ以上の経路をリボークする場合について、図14乃至図18のフローチャートを用いて、動作の一例を示す。

【0108】まず、リボーク情報入力部に経路 $p$ を入力(S1501)し、CPU205が $p$ を読み取り、作業メモリに格納(S1502)する。次に、CPU205が $p$ を境界集合計算部208に送る(S1503)。境界集合計算部208は、 $p$ の境界集合 $V'$ を計算(S1504)する。CPU205は、計算された $V'$ を受け取り作業メモリに格納(S1505)する。

【0109】CPU205は、 $p$ を随伴頂点集合計算部214に送る(S1506)。随伴頂点集合計算部214は、 $p$ の随伴頂点集合 $V$ を計算(S1507)する。CPU205は、その計算された $V$ を受け取り、作業メモリに格納(S1508)する。CPU205は、境界集合格納部から経路集合 $U'$ を読み出す(S1509)。CPU205は、差集合 $U' - V$ を求め作業メモリに格納(S1510)する。CPU205は、制御データ格納部から $U' - V$ に含まれない経路と、対応するリボケーション関数値とを削除する(S1511)。

【0110】次に、CPU205は、随伴頂点集合格納部から経路集合 $U$ を読み出す(S1512)。また、CPU205は作業メモリから $V'$ を読み出す(S1513)。CPU205は、差集合 $V' - U$ を求め、作業メモリに格納(S1514)する。CPU205が、 $V' - U$ に含まれる経路の数を数え、経路数を変数 $N$ に設定する(S1515)。その後、ここで、 $J$ を1に初期設定(S1516)する。

【0111】次に、 $J$ が $N$ より大きいとか否か判断し(S1517)、 $J$ が $N$ より大きくなるまで以下のステップS1518~S1528を繰り返す。

【0112】すなわち、

- ・CPU205が、 $V' - U$ の $J$ 番目の経路 $q[J]$ を経路関数計算部に送る(S1518)。
- ・経路関数計算部が経路関数値 $PF(q[J])$ を計算する(S1519)。
- ・CPU205が $PF(q[J])$ を受けとる(S1520)。
- ・CPU205は、マスター鍵 $K$ をマスター鍵格納部か

ら取得する (S1521)。

・CPU205が、数の組 (PF(q[J]), K) を暗号化部に送る (S1522)。

・暗号化部が、KをPF(q[J])で暗号化結果a(q[J])を得る (S1523)。

・CPU205は、a(q[J])を暗号化部から取得する (S1524)。

・CPU205が、経路と数の組 (q[J], a(q[J])) が制御データ格納部に存在するか否かを調べ (S1525)、存在すると判断した場合のみ、CPU205が、(q[J], a(q[J])) を制御データ格納部に追加 (S1527) する。

・Jを1増分する。 (S1528)。

の処理をJがNより大きくなるまで繰り返し行なう。

【0113】ステップS1517において、Jが大きくなったなら、次に、CPU205が、作業メモリ上の経路集合V'の要素数を数え、その数を変数Nに設定する (S1529)。そして、Jを再度1に設定し直す (S1530)。

【0114】次に、Jがnより大きいか否かを判断し (S1531)、大きくないならば、大きくなるまで、CPU205が、V'のJ番目の経路q[J]が境界集合格納部に存在するか否かを調べ (S1532)、存在しない場合のみq[J]を境界集合格納部に追加 (S1534) する。そして、jを1だけ増分し、ステップS1531に戻る。ステップS1531において、JがNより大きくなると、上記のステップS1529からステップS1535までと同様の処理を経路集合V'に対しても行なう (S1536～S1542)。そして終了する。

【0115】以上が、二つ以上の経路をリポークする場合の動作の一例である。これを以下に解説する。

【0116】二つ目以降のリポーク対象経路pを入力する時点で、制御データ生成装置20の境界集合格納部210と随伴頂点集合格納部212には、それぞれ現在の制御データを求める際に計算した境界集合 (の和集合) と頂点集合とが格納されている。同様に、制御データ格納部211には、現在の制御データが格納されている。

【0117】制御データ生成装置20は、先ずpの境界集合V'(p)と、pの随伴頂点集合V(p)とを求める。こ

$$C' = (C \cap (U' - V(p))) \cup (V'(p) - U) \cdots \text{式 8}$$

処理a～dの終了後、制御データ格納部は、C'に属する各経路と、当該経路におけるリポケーション関数値の組を格納している。処理e及びfは、次のリポーク対象経路の追加に備える為のデータ更新作業である。

【0119】P[1], ..., p[s]をリポーク対象経路とする。p[1], ..., p[s]に対して、順に上記処理a

$$\begin{aligned} X_s = & (V'(p[1]) \cup \cdots \cup V'(p[s])) - (V(p[1]) \cup \cdots \cup V(p[s])) \\ & \cdots \text{式 9.} \end{aligned}$$

【0120】各経路p[1], ..., p[s]を、境界集合

れらは、それぞれ境界集合計算部208と随伴頂点計算部214に経路pを送る事により求められる。CPU205は、V'(p)及びV(p)を作業メモリ206に格納する。次いで、制御データ生成装置20は以下の処理を順に行う。

a. 境界集合格納部210から経路の集合を読み出し、V(p)に含まれていない経路のみを作業メモリ206に書きこんで行く。境界集合格納部210に格納されている経路の集合をU' とすると、作業メモリ206には差集合U' - V(p)に属する経路が書きこまれる事になる。

b. 制御データ格納部211を調べ、作業メモリ206上の経路集合U' - V(p)に含まれていない経路と、当該経路におけるリポケーション関数値の組を消去する。

c. 随伴頂点集合格納部212から経路の集合Uを読み出す。作業メモリ206上の経路集合V'(p)の経路を順に選び、当該経路がUに含まれていなければ、当該経路を作業メモリ206にコピーする。この処理により、作業メモリ206上に差集合V'(p) - Uが得られる。

d. 作業メモリ206上の経路集合V'(p) - Uに属する各経路に関して、当該経路におけるリポケーション関数値を求める。そして、V'(p) - Uに属する各経路と、当該経路におけるリポケーション関数値の組を、制御データ格納部211に追加する。この際、重複する組は追加しない。

e. 境界集合格納部210の経路集合U'に、V'(p)に含まれる経路を追加する。この際、重複する経路は追加しない。これにより、U'の内容はU' ∪ V'(p)に変わる。

f. 随伴頂点集合格納部212の経路集合Uに、V(p)に含まれる経路を追加する。この際、重複する経路は追加しない。これにより、Uの内容はU ∪ V(p)に変わる。

【0118】上述の処理a～dを行う前の段階で、制御データ格納部211に格納されている経路の集合をCとおく。即ち、制御データ格納部211は、Cに含まれる各経路と、当該経路におけるリポケーション関数値の組を格納していたとする。a～dの処理を行う事により、制御データ格納部211の経路集合は、次の様に更新される。

～dを繰り返す事によって得られる制御データを (Xs, a(Xs)) とする。すなわち、当該制御データは、経路の集合Xsに属する各経路と、当該経路におけるリポケーション関数値の組の集合になっている。このとき、次式の成立が証明される。

計算部208と随伴頂点集合計算部214にそれぞれ入

力する事により、経路集合 $V'(p[1]), \dots, V'(p[s])$ と $V(p[1]), \dots, V(p[s])$ が得られるから、式9の右边を直接構成する動作も勿論可能であり、その様にしても良い。本実施の形態では、しかし、式9の右边を漸化的に構成する方法を採った。

【0121】具体例に、上記制御データの構成手順を適用してみる。デバイスID(2, 0, 1)と経路(1, 1)をリポーくする。まず、(2, 0, 1)をリポーくする制御データは、次の通り。

((0), a(0))((2, 1), a(2, 1))((2, 0, 0), a(2, 0, 0))((1), a(1))((2, 2), a(2, 2))((2, 0, 2), a(2, 0, 2))

この段階で、境界集合格納部210に格納されている経路集合 $U'$ は次の通りである。

((0)(2, 1)(2, 0, 0)(1)(2, 2)(2, 0, 2)

また、随伴頂点集合格納部212は、次の経路集合 $U$ を格納している。

((2)(2, 0)(2, 0, 1)

さて、経路(1, 1)を境界集合計算部208に入力する事により、下記の境界集合 $V'(1, 1)$ を得る。

((0)(1, 0)(1, 2)

また、経路(1, 1)を随伴頂点集合計算部に入力する事により、下記の随伴頂点集合 $V(1, 1)$ を得る。

((1)(1, 1)(1, 1, 0)(1, 1, 1)(1, 1, 2)

上記処理aで作業メモリに書きこまれる経路集合 $U' - V(1, 1)$ は、従って、次の様になる。

((0)(2, 1)(2, 0, 0)(2, 2)(2, 0, 2)

処理bにより、制御データ格納部の内容は次のように更新される。

((0), a(0))((2, 1), a(2, 1))((2, 0, 0), a(2, 0, 0))((2, 2), a(2, 2))((2, 0, 2), a(2, 0, 2))

処理cの結果作業メモリに格納される経路集合 $V'(1, 1) - U$ は次の様になる。

((0)(1, 0)(1, 2)

処理dの結果、制御データ格納部の内容、即ち制御データは次のように変化する。

((0), a(0))((2, 1), a(2, 1))((2, 0, 0), a(2, 0, 0))((2, 2), a(2, 2))((2, 0, 2), a(2, 0, 2))((1, 0), a(1, 0))((1, 2), a(1, 2))

このデータが、制御データとして出力される。

【0122】更に、処理eにより、境界集合格納部の内容は次の様に更新される。

((0)(2, 1)(2, 0, 0)(1)(2, 2)(2, 0, 2)(0)(1, 0)(1, 2)

最後に処理fにより、随伴頂点集合格納部の内容は次の様に変化する。

((2)(2, 0)(2, 0, 1)(1)(1, 1)(1, 1,

0)(1, 1, 1)(1, 1, 2)

なお、処理dの結果更新された制御データ格納部が格納する経路集合は、確かに下記経路集合に一致する。

((V'(2, 0, 1) ∪ V'(1, 1)) - (V(2, 0, 1) ∪ V'(1, 1)))

次に、随伴頂点集合計算部214の動作の一例を図19及び図20に示し、説明する。

【0123】初めに、随伴頂点集合計算部214は、経路 $p$ を受け取り(S1601)、経路 $p$ の長さを変数 $L$ に格納する(S1602)。

【0124】次に、デバイスID長格納部からデバイスID長を読み取り、変数 $N$ に格納する(S1603)。

【0125】そして、 $N-L$ が0より大きいか否かの判定を行なう(S1604)。 $N-L$ が0より大きければ、 $J$ を1に設定する(S1605)。その後、 $J$ が $L$ より大きくなるまで、 $J$ を1ずつ増分しながら経路( $p(1), \dots, p(J)$ )を出力する(S1606～S1608)。

【0126】一方、ステップS1604にて、 $N-L$ が0より大きくなければ、以下の処理に進む。まず $J$ を1に設定する(S1609)。その後、 $J$ が $N-L$ より大きくなるまで、 $J$ を1ずつ増分(S1618)しながら、以下の処理を行う。

・ $J$ 個の成分を持つ整数配列 $X, Y$ を用意する(S1611)。

・ $Y[0] = 2Y[J-1] = 2$ とする(S1612)。

・ $x$ を0に設定する(S1613)。

・ $x = Y$ となるまで、 $X$ を1ずつ増分しながら $x$ の三進表現を配列 $X$ に格納し、経路( $p(1), \dots, p(L)$ ),  $X[0], \dots, X[J-1]$ )を出力する(S1614～S1617)。

なお、ステップS1610において、 $J$ が $N-L$ より大きくなった際に処理を終了する。

【0127】上記で説明した動作について、以下に解説する。

【0128】随伴頂点集合計算部214には、図示しない予めデバイスIDの長さが格納されているデバイスID長格納部を持っている。随伴頂点集合計算部214は、格納されたデバイスIDの長さ取得し、読みこんだ経路の長さと比較する。読みこんだ経路の長さ $L$ がデバイスIDの長さ $N$ より小さい場合、デバイスIDの長さ $N$ と経路の長さの差をとり、その差 $N-L$ に一致する要素数を持つ整数配列 $X, Y$ を用意する。

【0129】更に、 $Y$ の全ての成分を2に設定する。 $Y$ は終了条件判定用である。整数の変数 $x$ を0から $3N-L-1$ まで変化させながら、 $x$ の三進表現を配列 $X$ に格納して行く。 $X$ は0, ..., 0( $N-L$ 個)から、2, ..., 2( $N-L$ 個)まで変化する。

【0130】随伴頂点集合計算部214は、各 $x$ の値毎に、( $p(1), \dots, p(L)$ ),  $X[0], \dots, X[N-L]$ )を出力する。 $X$ は、随伴頂点集合の要素の中で、ワイ

ルド・カードを用いて表現される要素を生成する。リボーク対象経路 $p$ の長さがデバイスIDの長さより小さいのは、 $(p(1), \dots, p(L), ?, \dots, ?)$ と表現される一群のデバイスIDを一括してリボークする場合及び、その場合に限られる。この種の一括リボケーションは、特定の製造業者による特定分野の製品を全てリボークする場合等に発生する。

【0131】一度にリボークされるデバイスIDの数は、従って、高々数百～数千万のオーダーである。そして、生成すべき随伴頂点集合の個数は、リボーク対象デバイスIDの数の約二倍である。この数は大きい、しかし、この程度の数の随伴頂点集合を生成する処理は、現行の計算機で現実的な時間内に終了させる事ができる。しかも、この処理は、リボーク対象デバイス群が新たに追加された場合に、リボーク主体が保有する制御データ生成装置20で一度だけ実行すれば良い性質のものである。

【0132】次に、境界集合計算部208の動作を図21に示し説明する。

【0133】境界集合計算部208は、経路 $p$ を受け取って(S1701)この経路 $p$ の差が差を変数 $L$ に格納する(S1702)。Jを1に設定し(S1703)、JがLより大きくなるまで、次の処理を繰り返す。  
 ・Iに0を設定する(S1705)。  
 ・Iが3より小さいか否かを判定し(S1706)、小さければ、Iは $P(J)$ と同じかを判定し(S1707)、同じでなければ、経路 $(p(1), \dots, p(J-1), I)$ を出力し(S1708)、Iを1だけ増分して、再度、ステップS1706の判定処理を行う。  
 ・ステップS1706の判定処理で小さくないと判断されるとJを1増分する。なお、JがLより大きくなる終了となる。

【0134】上記動作を換言すると、境界集合計算部208は、受け取った経路 $p$ について、経路 $p$ の最後の数字 $p(L)$ を、 $p(L)$ とは異なった数字に置き換えた経路を作り、それを出力する。経路の長さがLで、経路を構成する数字の範囲が0から $m-1$ であるとき、境界集合に含まれる経路の総数は $mL$ である。

【0135】本発明の制御データ生成装置20は、従来のMK B技術におけるMK B生成装置と対比すべきものである。本発明の制御データ生成装置20は、随伴頂点集合計算部214を具備する点に特徴がある。また、本発明の制御データ生成装置20は、境界集合計算部208を具備する点に特徴がある。また、本発明の制御データ生成装置20は、経路関数計算部213を具備する点に特徴がある。随伴頂点集合計算部：数字の並びである経路について、入力された経路に属する数字の一部または全部を用いて作られる部分経路、または当該経路に少なくとも一つの数字を追加して作られる経路を、少なくとも一つ出力する。境界集合計算部：入力された経路に属する数字の一部または全部を用いて作られる部分経路

の少なくとも一つについて、当該部分経路に属する数字の一部または全部を変更して作られる経路を出力する。

【0136】また、本実施例の制御データ生成装置20は、以下のデータを出力する点に特徴がある。

一数字の並びである経路と、当該経路に対応付けられた数値を、少なくとも一組出力する。

一特に、上記経路は、境界集合計算部の出力経路である。

一特に、上記経路に対応付けられた数値は、当該経路におけるリボケーション関数値である。

一リボケーション関数値：数字の並びである経路に対して定められる数値で、当該経路における経路関数値とマスター鍵に依存する数値である。

【0137】次にメディア上の制御データについて説明する。

【0138】制御データ生成装置20の出力である制御データは、ネットワークや放送を通じて利用デバイスに配布される。あるいは、メディアに記録されて配布される事もある。制御データがメディアに記録されて配布される場合、制御データは、例えば、経路と当該経路におけるリボケーション関数値のペアを列挙する形式で、メディアに記録される。この場合、経路を数字の列として、辞書式に順序づけて列挙するようにしても良い。

【0139】図22に一例を挙げる。経路はPDに格納される。PDは8ビットのNLフィールドとPTフィールドを持つ。NLフィールドには、経路を構成する数字の数が記録されている。その数を $v$ とする。PTフィールドは $2 \times v$ ビットの長さを持つ。本実施例において、経路を構成する数字は0, 1または2であるから、それは2ビットで表現される。例えば、経路(2, 0, 1)の場合、NLフィールドの値は3であり、PTフィールドはこの6ビット"10001"のビット列である。

【0140】BPは、経路と、当該経路におけるリボケーション関数値の組を格納する。BPはPDフィールドとVRフィールドから構成される。PDフィールドについては既に説明した。VRフィールドは128ビットの符号なし整数である。制御データのフォーマットはCDと書かれているものである。CDフォーマットは、NBPフィールドと0個以上のBPフィールドから構成される。NBPフィールドには、制御データを構成するBPフィールドの数が書かれている。その数を $n$ とすると、制御データは、BP 1からBP  $n$ まで $n$ 個のBPフィールドを持つ。

【0141】本発明の制御データが記録されたメディアは、少なくとも下記の情報を含む点に特徴がある。

一数字の並びである経路と、当該経路に対応付けられた数値が、少なくとも一組含まれる。当該経路と数値の組は、本発明の制御データ生成装置20の出力である。

一特に、上記経路は、本発明の制御データ生成装置20における境界集合計算部208が出力した経路である。

一特に、上記経路に対応付けられた数値は、当該経路に

おけるリボケーション関数値である。

【0142】次に利用デバイスについて説明する。

【0143】利用デバイスは、データを利用する為にマスター鍵を必要とする。それは、例えばデータがデータ・キーで暗号化されており、更にデータ・キーがマスター鍵で暗号化されているからである。あるいは、マスター鍵でデータが暗号化されている場合も考えられる。各利用デバイスは、デバイス情報を割り当てられている。デバイス情報は、本実施例のデバイス情報生成装置の出力である。即ち、デバイスIDと利用デバイスは制御データを読み込み、リボケーション関数値を復号して、マスター鍵を取得しようとする。

【0144】図23に利用デバイスの構成を示す。

【0145】利用デバイスの動作を、図24、図25に示し以下に説明する。

【0146】まず、制御データ入力部に制御データ入力する(S1801)。CPUは制御データを制御データ入力部から読み取る(S1802)。CPUは制御データを制御データ格納部に格納する(S1803)。CPUは、デバイスIDの長さをデバイス情報格納部から読み、変数Nに設定する(S1804)。ここで、Jに1を初期設定する(S1805)。

【0147】この後、リボーク判定処理を行う。

【0148】まず、JはNより大きいかなかを判別する(S1806)。大きければ、リボークであり、CPUは、メッセージ表示部に当該利用デバイスのリボークを表示する(S1811)。一方、現段階でJはNより大きくなければ、次の処理を行う。

・CPUが、デバイスIDの部分経路(P[1], , P[J])をデバイス情報格納部から読み出す(S1807)。

・CPUが、経路(P[1], , P[J])を制御情報格納部から検索する(S1808)。

・検索した結果、当該経路が存在しない場合には、Jを1だけ増分し、ステップS1806に戻る(S1809、S1810)。

【0149】ステップS1809にて当該経路が存在したと判断されると、CPUは、経路(P[1], , P[J])におけるリボケーション関数値RV[J]を読む(S1812)。次に、CPUは、J番目の経路関数値PF[J]をデバイス情報格納部から読む(S1813)。そし

$$\begin{aligned} &((0), a(0))((2, 1), a(2, 1))((2, 0, 0), a(2, 0, 0)) \\ &((2, 2), a(2, 2))((2, 0, 2), a(2, 0, 2)) \\ &((1, 0), a(1, 0))((1, 2), a(1, 2)) \cdots \text{式10} \end{aligned}$$

制御データ生成装置20の説明で示したように、この制御データは、デバイスID(2, 0, 1)と経路(1, 1)とをリボークするものである。

【0157】今、利用デバイス30が、デバイスID(1, 0, 2)を有していたとする。利用デバイス30のCPU307は、先ず部分経路(1)を、制御データ格納部302に格納されている、式10の制御データから探

て、CPUは、数の組(PF[J], RV[J])を復号部に送る(S1814)。

【0150】復号部では、PF[J]でRV[J]を復号し、得られた結果RをCPUに返す(S1815)。CPUは、Rをデータ利用部に送る(S1816)。CPUは、データをデータ入力部から読む(S1817)。CPUは、データをデータ利用部に送る(S1818)。そして、データ利用部がデータを利用することとなる(S1819)。

【0151】以上の動作を解説する。

【0152】デバイス情報格納部303には、本実施例のデバイス情報生成装置10が生成したデバイス情報の一組が格納されている。即ち、デバイスIDと、経路関数値の組である。

デバイスID: (P[1], P[2], P[3])

経路関数値: (PF1, PF2, PF3)

デバイス情報格納部303は、このデバイス情報に加えて、デバイスIDの長さを格納している。本実施例の場合、デバイスIDの長さは3である。

【0153】利用デバイス30は、データ利用に先立って、制御データを読み込み、制御データ格納部302に格納する。CPU307は、デバイスIDの部分経路を順に取り出し、制御データに含まれる経路の中で当該経路に一致するものがあるかを調べる。一致する経路が存在すれば、当該経路に対応するリボケーション関数値を制御データ格納部302から読み出す。

【0154】次いで、CPU307は、デバイス情報格納部303から当該部分経路に対応する経路関数値を読み出し、経路関数値とリボケーション関数値の組を復号部306に送る。

【0155】復号部306は、経路関数値を鍵として、リボケーション関数値を復号し、その結果RをCPU307に返す。Rの値は、リボーク主体がマスター鍵として定めた値Kに等しい。CPU307は、データ利用部304にRの値を送り、次いでデータ入力部305から読みこんだデータを、データ利用部304に送る。

【0156】利用デバイス30によって読みこまれた制御データは、例えば、次の様に制御データ格納部に格納される。

す。この経路は式10の制御データ中に存在しない。従って、利用デバイス30のCPU307は、次の部分経路(1, 0)を式10の制御データ中から探す。この経路は存在する。利用デバイス30のCPU307は、対応するリボケーション関数値a(1, 0)を制御データ格納部302から読み出し、当該関数値を経路関数値PF2 = PF(1, 0)と共に復号部に送る。復号部306は、数値



PF2を鍵として数値 $a(1, 0)$ を復号し、その結果 $R$ をCPU307に返す。先述の様に、 $a(1, 0)$ は $\text{Enc}(PF(1, 0), K)$ に等しい。従って、復号結果 $R$ はマスター鍵 $K$ に等しい。データ利用部304は、 $K$ の値を得て、データ利用を行う事ができる。

【0158】一方、利用デバイス30が、例えば(1, 1, 2)と言うデバイスIDを有しているとする。

【0159】CPU307は、先ず部分経路(1)を、式10の制御データから探す。この経路は式10の制御データ中に存在しない。従って、CPUは次の部分経路(1, 1)を探す。この経路もまた式10の制御データ中には存在しない。CPU307は、更に部分経路(1, 1, 2)を、式10の制御データ中から探す。この経路もまた存在しない。部分経路の長さがデバイス長に一致するまで検索を行ったので、CPU307は当該利用デバイス30がリポーカされている事をメッセージ表示部308に表示し、処理を終了する。

【0160】本実施例の利用デバイスは、以下の点に特徴がある。

1. デバイス情報格納部の特徴：デバイス情報格納部には、デバイスIDの一部または全部の数字をとって作られる部分経路に対応する数値である経路関数値が少なくとも一つ予め格納されている。或いは、デバイス情報格納部は、本実施例のデバイス情報生成装置が生成したデバイス情報を格納している。

2. 入力データの特徴：入力データは制御データを含む。制御データは、経路と当該経路に対応するリボケーション関数値を、少なくとも一つ含む。

3. 動作の特徴：

(ア) 入力された制御データに含まれる経路の中から、デバイスIDの一部または全部の数字を並べて得られる部分経路に一致する経路を検索する。

(イ) 上記3. (ア)の検索の結果一致する経路が見つからなかった場合、当該利用デバイス30がリポーカされていると判断し、リポーカ時の処理を行う。

(ウ) 上記3. (ア)の検索の結果一致する経路が見つかった場合、制御データから当該経路に対応するリボケーション関数値を読み出し、デバイス情報格納部303に当該経路に対応づけて格納されている経路関数値を鍵として復号する。上記特徴は、いずれもMKB技術における利用機器には無いものである。

【0161】(第2の実施の形態) 次に、第2の実施の形態を説明する。

【0162】第1の実施の形態における制御データ生成装置20は、随伴頂点集合計算部214を具備しており、制御データを作る際に、リポーカ対象経路の随伴頂点集合を実際に生成した。

【0163】しかし、制御データ生成の為に、任意の経路がある経路の随伴頂点集合に属するか否かを判定する随伴頂点集合判定部があれば十分である事が、制御デ

ータ生成装置20における制御データの生成手続きを検討する事により分かる。この事を以下に説明する。

【0164】第1の実施の形態の制御データ生成装置20の手続きaでは、境界集合格納部が格納する経路集合 $U'$ と、経路 $p$ の随伴頂点集合 $V(p)$ の差集合が計算される。具体的には、 $U$ に含まれる経路 $q$ を順に選び、 $q$ が $V(p)$ に含まれていない場合に限り、 $q$ を作業メモリに記録して行けば良い。即ち、必ずしも随伴頂点集合 $V(p)$ を実際に生成する必要は無く、 $q$ が $V(p)$ に含まれるか否かが分かれば良い。

【0165】第1の実施の形態の制御データ生成装置20の手続きcでは、随伴頂点集合格納部212が格納する経路集合 $U$ と、経路 $p$ の境界集合 $V'(p)$ の差集合 $V'(p) - U$ を求める。この差集合を求める際は、 $V'(p)$ の各経路が $U$ に含まれるか否かが分かれば良い。その為に、必ずしも $U$ の要素を全て生成する必要は無い。

【0166】リポーカ対象経路がデバイスIDの長さより相当に短い場合、当該経路の随伴頂点集合は、既に述べた様に、かなり大きな集合になる。この集合を生成する手間を省くことは、制御データ生成装置20の動作を効率化する上で重要である。この点を改良した制御データ生成装置40の構成を図26に示す。

【0167】また、制御データ生成装置40の初回の動作を図27乃至図29に、更新時の動作を図30乃至図34に示し説明する。

【0168】まず、初回の動作であるが、基本的には制御データ生成装置20の初回の動作(図11乃至図13)とほぼ同様な動作を行う。

【0169】違いは、図13におけるステップS1421～S1423の処理が無い点、及びステップS1424が、 $V$ を随伴頂点集合格納部に格納するのに対し、ステップS1921では、 $p$ を随伴頂点集合格納部に格納する点である。このように制御データ生成装置40の初回の動作は、制御データ生成装置40の初回の動作に比較し、簡略化可能となった。

【0170】次に、更新時の動作について説明する。更新時においても制御データ生成装置40の更新時の動作と、類似している。従って、違いについてのみ以下に書き出す。

【0171】図30に記載される動作に関しては、ステップ2006において、制御データ生成装置20では、 $p$ を随伴頂点集合計算部に設定していたのを、制御データ生成装置40では、 $p$ を随伴頂点集合判定部に設定することとなる。また、ステップS1507、S1508に相当する処理が不要になっている。

【0172】図31に記載される動作に関しては、ステップS2011「CPUが $U$ を随伴頂点集合判定部に設定する」点が追加されたのみである。

【0173】図32及び図33に記載される動作に関しては、制御データ生成装置20の図16及び図17に示

される動作と同様な動作で実現される。

【0174】図34に示される動作が制御データ生成装置20での動作と異なっている。

【0175】すなわち、CPUは、作業メモリから経路pを読み出し(S2035)、随伴頂点集合格納部にpが存在するか否かを調べる(S2036)。pが存在すれば、そのまま終了し、pが存在しなければ、pを随伴頂点集合格納部に追加し(S2038)、動作を終了する。

【0176】随伴頂点集合判定部416は、図30に示されている動作において、差集合 $U' - V$ を求める際に用いられる。ここにVは、リボーク対象経路pの随伴頂点集合を表す。

【0177】また、随伴頂点集合判定部416は、図31に示されている動作において、差集合 $V' - U$ を求める際にも用いられる。

【0178】図35は、差集合 $U' - V$ を求める動作を説明したものであり、図36は差集合 $V' - U$ を求める動作を説明したものである。これらの動作は、同様な動作を行うので、図35に従って説明し、図36の説明は、省略する。

【0179】まず、CPUが $U'$ に属する経路の数を変数Nに設定する(S2101)。また、Jを1に設定する(S2102)。

【0180】その後、JがNより大きくなるまでの次の処理を繰り返し行う。

- ・CPUは、 $U'$ のJ番目の経路q[J]を随伴頂点集合判定部に送る(S2104)。

- ・随伴頂点集合判定部は、判定動作を行って、判定結果RをCPUへ送る(S2105)。

- ・この判定結果Rが0でない場合のみ、CPUは、経路q[J]を $U' - V$ の経路として、作業メモリに格納する(S2107)。

- ・Jを1だけ増分する(S2108)。

【0181】次に、図37及び図38に随伴頂点集合判定部416の動作を示す。随伴頂点集合判定部416の動作は、経路集合設定動作と、判定動作に分けられる。

【0182】経路集合設定動作は、図37の通り、経路集合Pを受け取り(S2301)、Pを経路集合格納部に格納する(S2302)。

【0183】判定動作は、図38に示したように、まず、判定対象経路pを受け取る(S2401)。次に、経路集合格納部に格納されている経路集合Pの要素数を変数Nに設定する(S2402)。Jを1に設定する(S2403)。JがNより大きくなるか(S2404)、IがLより大きくなるか(S2408)まで、以下の処理を行う。

- ・PのJ番目の経路q[J]の長さsと経路pの長さとのうち、大きくない方の長さを変数Lに設定する(S2406)。

- ・Iに1を設定する(S2407)。

- ・IはLより大きいかなを判定する(S2408)。

- ・経路q[J]とpのI番目の数字を比較し(S2409)。

- ・一致するか否かを判定する(S2410)。

- ・一致した場合、Iを1増分し(S2411)、ステップS2408へ戻る。一致しなければ、Jを1増分し(S2412)、ステップS2404へ戻る。

【0184】ステップS2404において、JがNより大きいと判断されると、0を出力し(S2405)、終了する。または、ステップS2408において、IがLより大きいと判断されると、1を出力し(S2413)、終了する。

【0185】以上の判定動作を換言すると、判定動作は、判定対象経路を受け取り、当該経路が、経路集合格納部に格納されている経路の随伴頂点集合になっているかどうかを判定するものである。経路集合設定動作は、経路集合格納部の内容を設定する動作である。

【0186】二つの経路p, qを考える。経路qの長さが経路pの長さより長くない場合、qがpの随伴頂点集合に含まれる事と、qの数列全体がpの数列の一部に一致する事とは同値である。

【0187】経路qの長さが経路pの長さより長い場合は、pの数列全体がqの数列の一部に一致する事と、qがpの随伴頂点集合に含まれる事とは同値である。随伴頂点集合判定部416は、経路集合格納部417から経路qを順に選択し、qの数の並びと入力された経路の数の並びとを比較する事により、入力された経路がqの随伴頂点集合に含まれるかなを判定している。

【0188】本実施例の制御データ生成装置40は、随伴頂点集合判定部416を具備する点に特徴がある。一随伴頂点集合判定部416：数字の並びである経路を少なくとも一つ設定することができる。入力された経路が、設定された経路に属する数字の一部または全部を用いて作られる部分経路に一致するか否かを判定する。或いは、入力された経路が、設定された経路に少なくとも一つの数字を追加して作られる経路に一致するか否かを判定する。そして、判定結果を出力する。

【0189】

【発明の効果】本発明によれば、利用デバイスにおけるデータの利用を、制御データによって制御する技術において、重大なセキュリティ上の問題と、リボケーションに際して発生し、ユーザーの利便性を大きく損なう副作用の問題とを、共に回避する事が可能である。

【図面の簡単な説明】

【図1】リボケーション木構造の一

【図2】リボケーション木構造の二

【図3】リボケーション木構造の三

【図4】デバイス情報生成装置

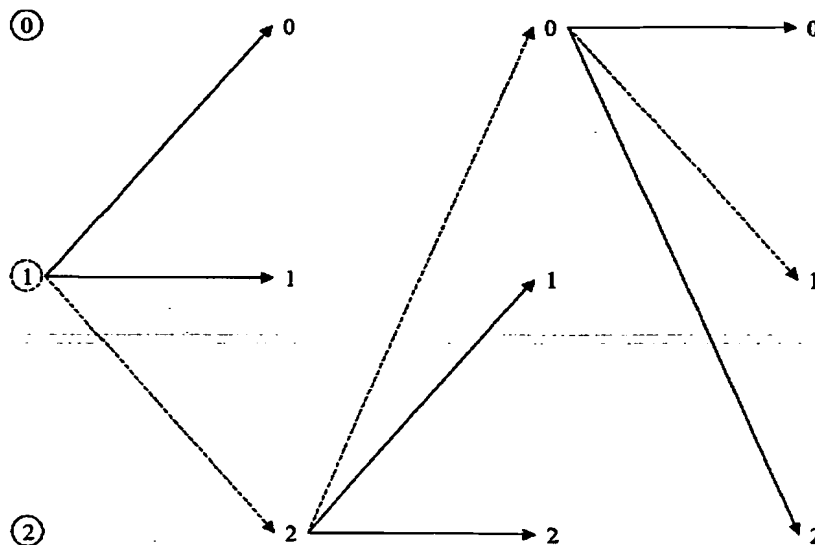
【図5】経路関数計算部の動作

【図6】デバイス情報生成装置の動作の第一

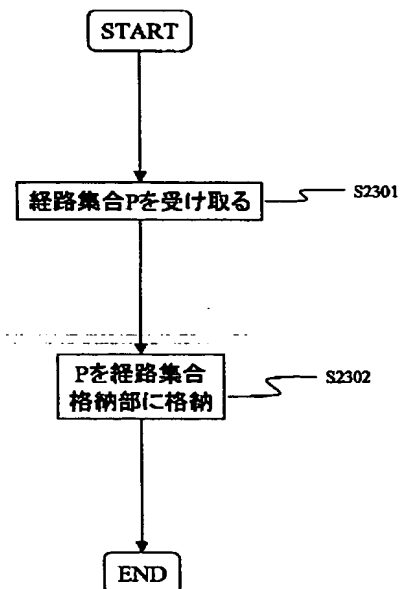
【図 7】 デバイス情報生成装置の動作の第二  
 【図 8】 デバイス情報生成装置の動作の第三  
 【図 9】 制御データ生成装置の構成  
 【図 10】 制御データ生成装置の動作（初期状態）  
 【図 11】 制御データ生成装置の動作（初回）の第一  
 【図 12】 制御データ生成装置の動作（初回）の第二  
 【図 13】 制御データ生成装置の動作（初回）の第三  
 【図 14】 制御データ生成装置の動作（更新時）の第一  
 【図 15】 制御データ生成装置の動作（更新時）の第二  
 【図 16】 制御データ生成装置の動作（更新時）の第三  
 【図 17】 制御データ生成装置の動作（更新時）の第四  
 【図 18】 制御データ生成装置の動作（更新時）の第五  
 【図 19】 随伴頂点集合計算部の動作の第一  
 【図 20】 随伴頂点集合計算部の動作の第二  
 【図 21】 境界集合計算部の動作  
 【図 22】 制御データのフォーマット  
 【図 23】 利用デバイスの構成  
 【図 24】 利用デバイスの動作の第一  
 【図 25】 利用デバイスの動作の第二  
 【図 26】 制御データ生成装置の構成  
 【図 27】 制御データ生成装置の動作（初回）の第一  
 【図 28】 制御データ生成装置の動作（初回）の第二  
 【図 29】 制御データ生成装置の動作（初回）の第三  
 【図 30】 制御データ生成装置の動作（更新時）の第一  
 【図 31】 制御データ生成装置の動作（更新時）の第二

【図 32】 制御データ生成装置の動作（更新時）の第三  
 【図 33】 制御データ生成装置の動作（更新時）の第四  
 【図 34】 制御データ生成装置の動作（更新時）の第五  
 【図 35】 差集合  $U' - V$  を求める動作  
 【図 36】 差集合  $V' - U$  を求める動作  
 【図 37】 随伴頂点集合判定部の動作（設定時）  
 【図 38】 随伴頂点集合判定部の動作（判定時）  
 【図 39】 MKB処理  
 【図 40】 デバイス情報生成装置  
 【図 41】 デバイス情報生成装置の動作  
 【図 42】 デバイス情報生成装置の動作  
 【図 43】 MKB生成装置の構成  
 【図 44】 メディア・キー・ブロック生成装置の動作（生成時）の第一  
 【図 45】 メディア・キー・ブロック生成装置の動作（生成時）の第二  
 【図 46】 メディア・キー・ブロック生成装置の動作（更新時）  
 【図 47】 利用機器の構成  
 【図 48】 利用機器の動作の第一  
 【図 49】 利用機器の動作の第二  
 【符号の説明】  
 10・・・デバイス情報生成装置、20・・・制御データ生成装置  
 40・・・制御データ生成装置

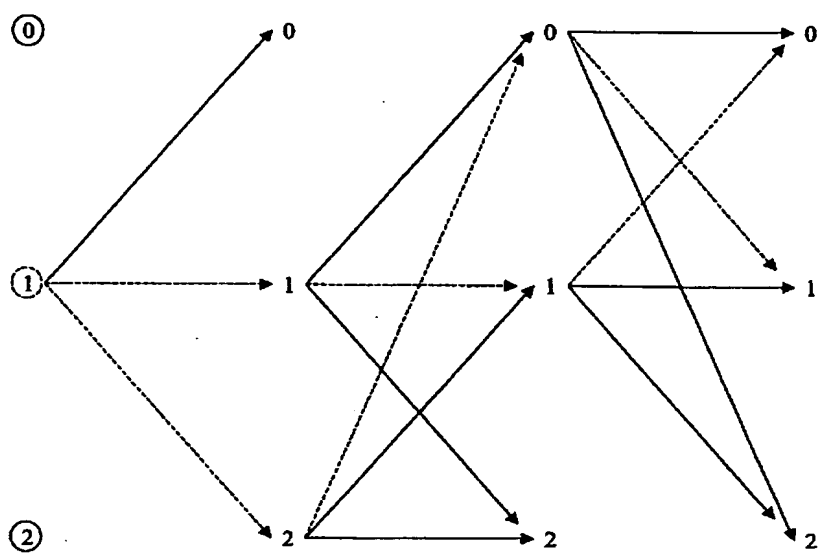
【図 1】



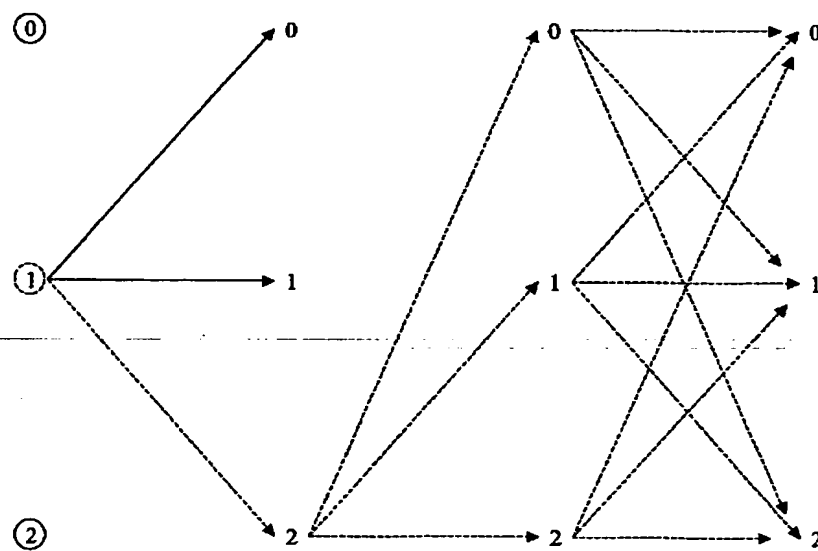
【図 37】



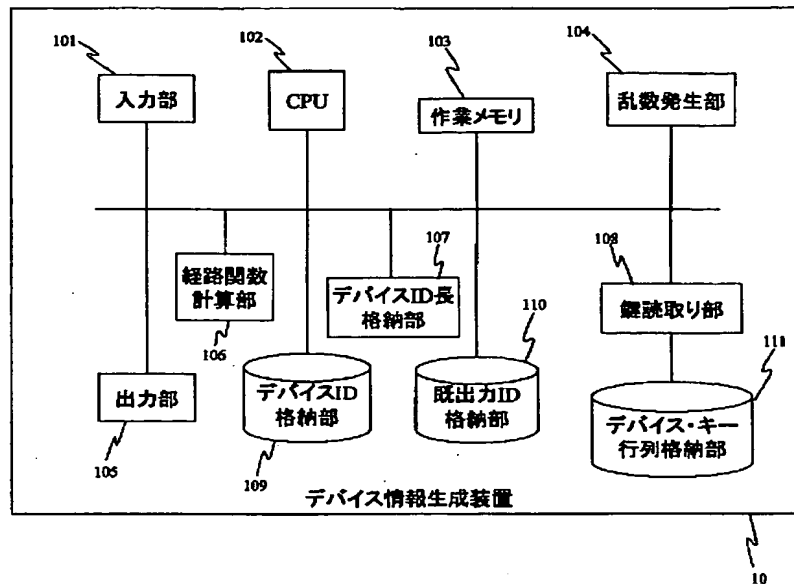
【図2】



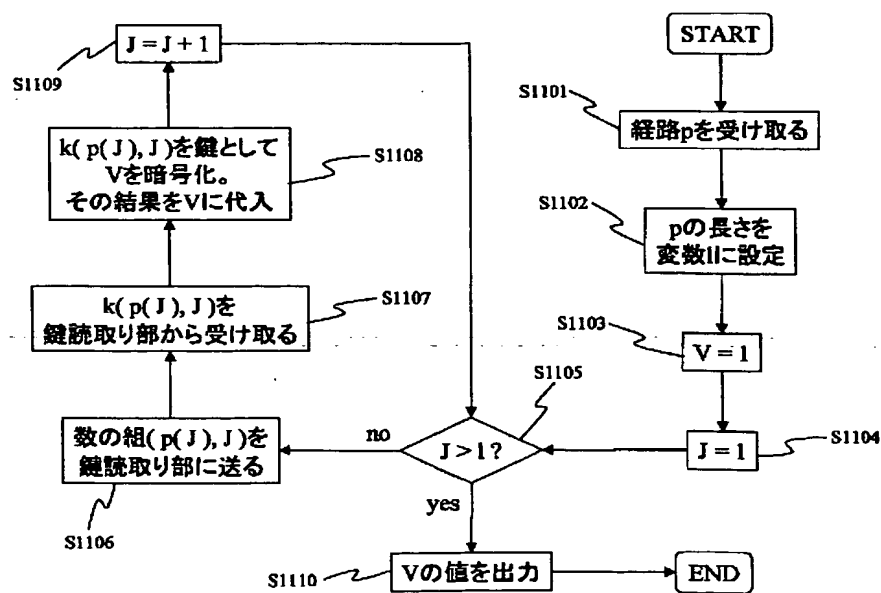
【図3】



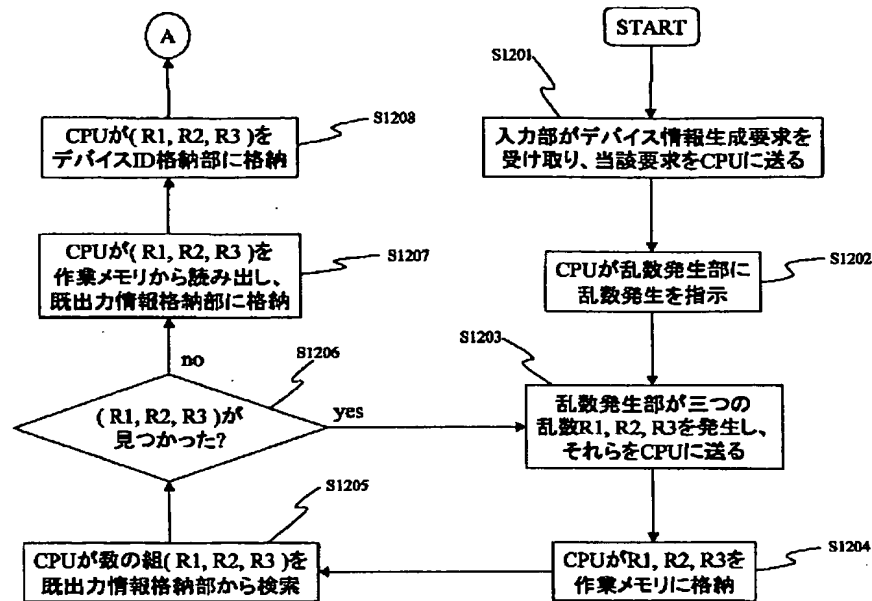
【図4】



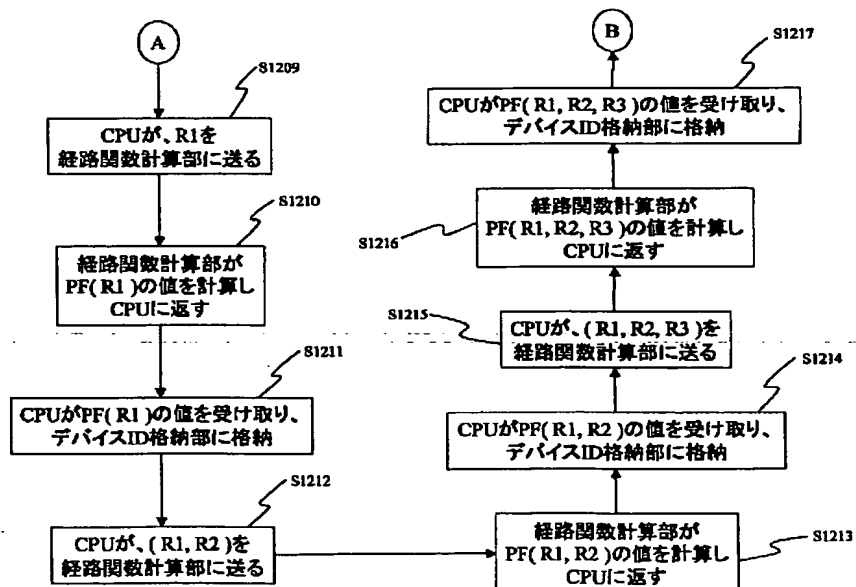
【図5】



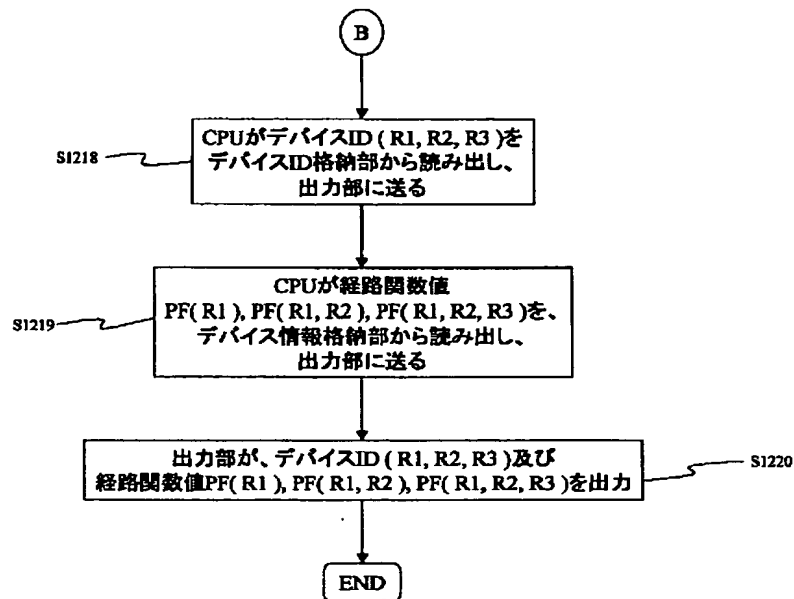
【図6】



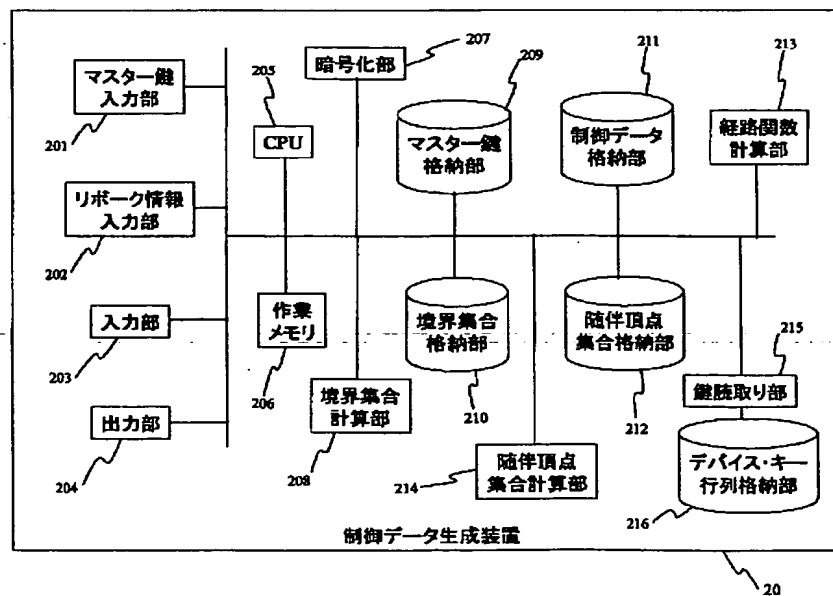
【図7】



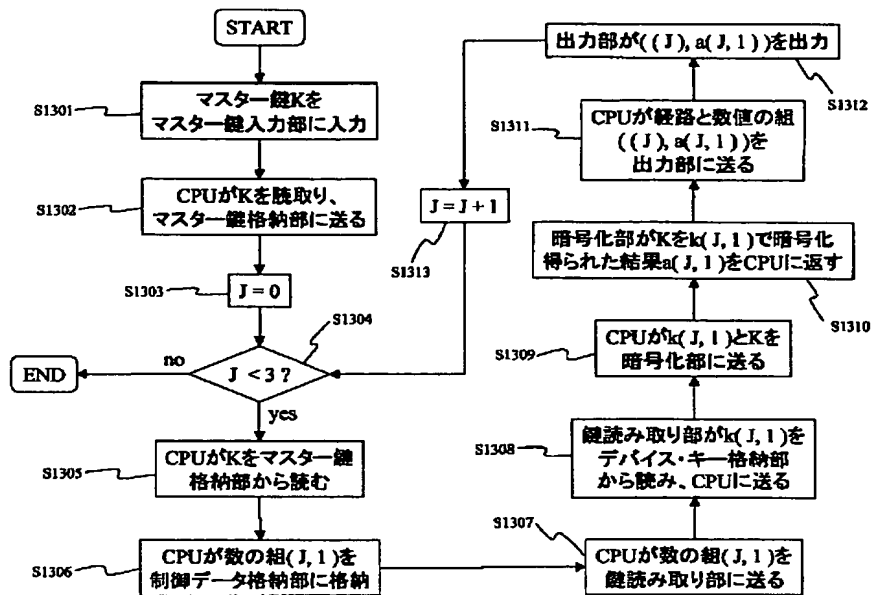
【図8】



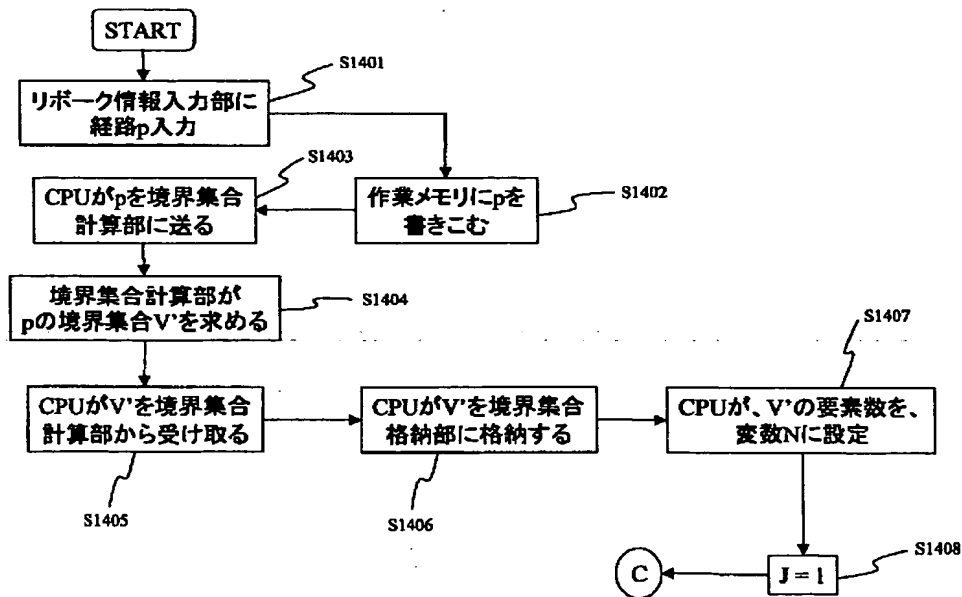
【図9】



【図10】

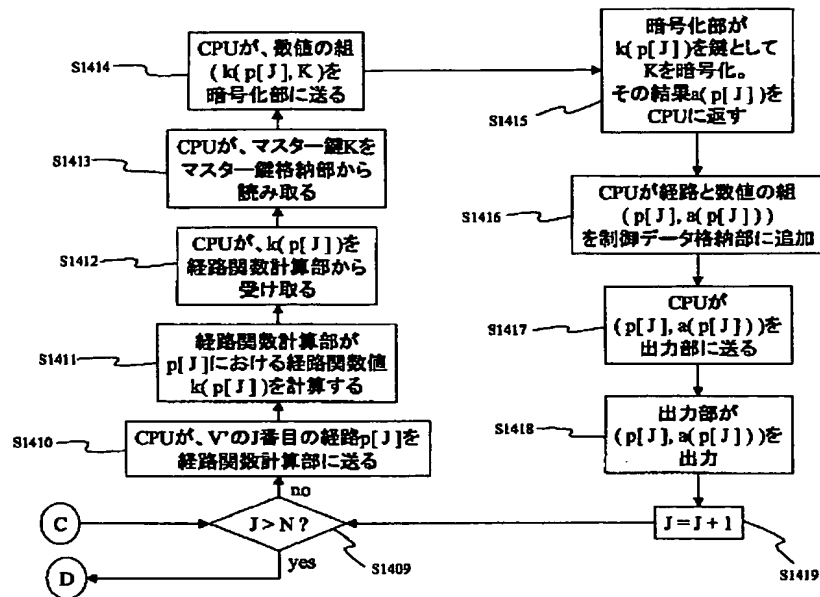


【図11】

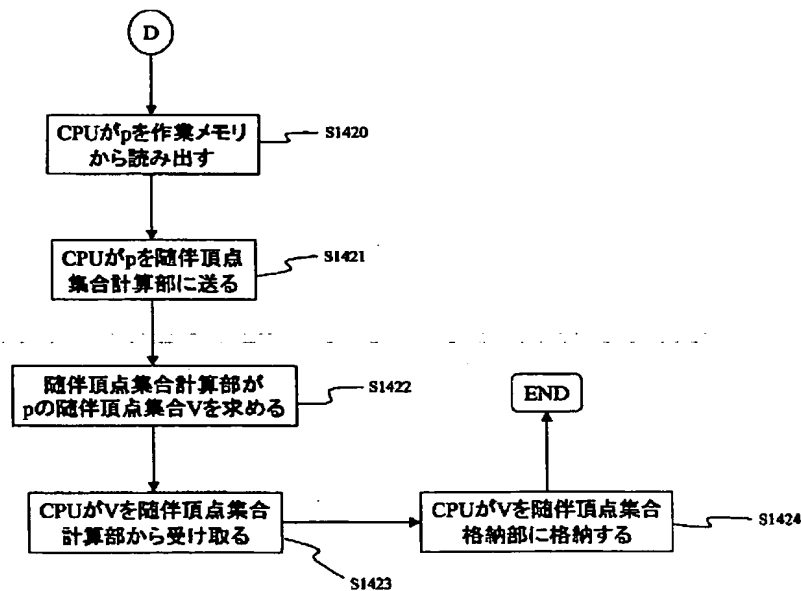




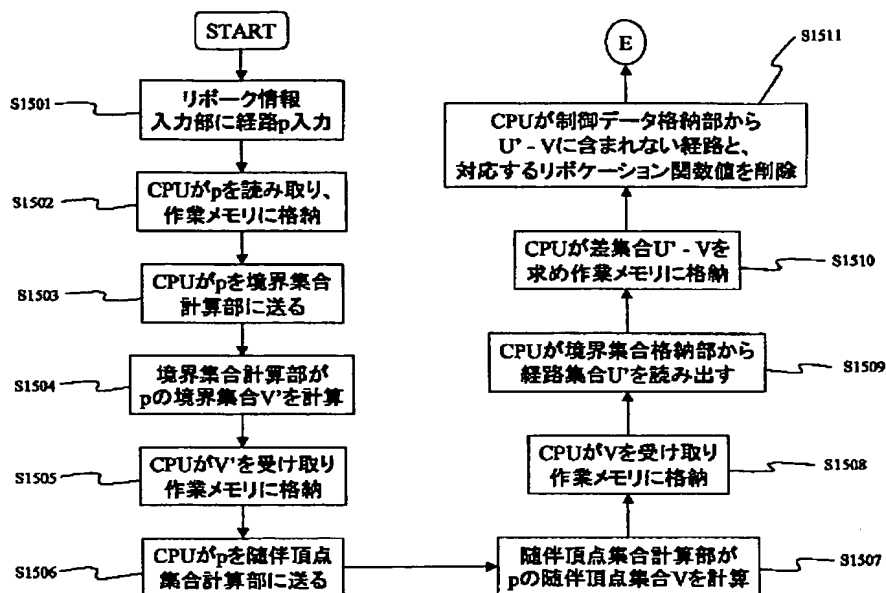
【図12】



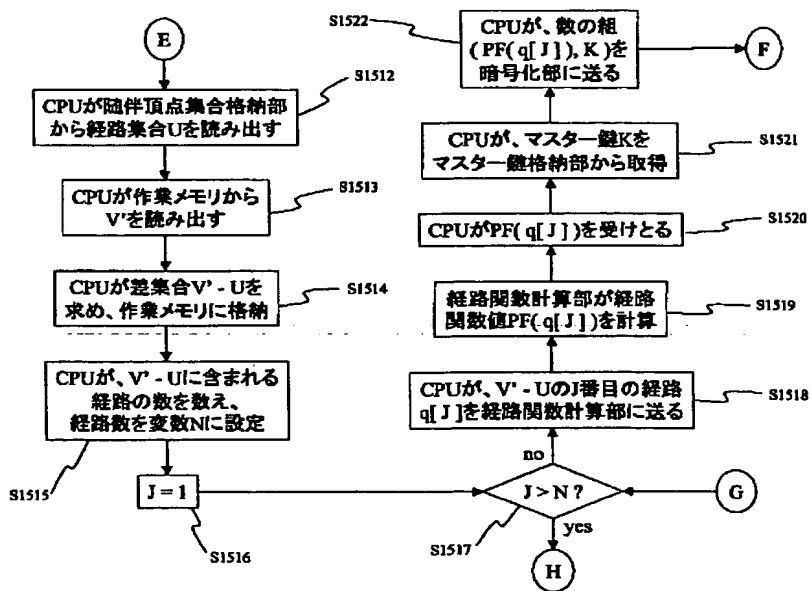
【図13】



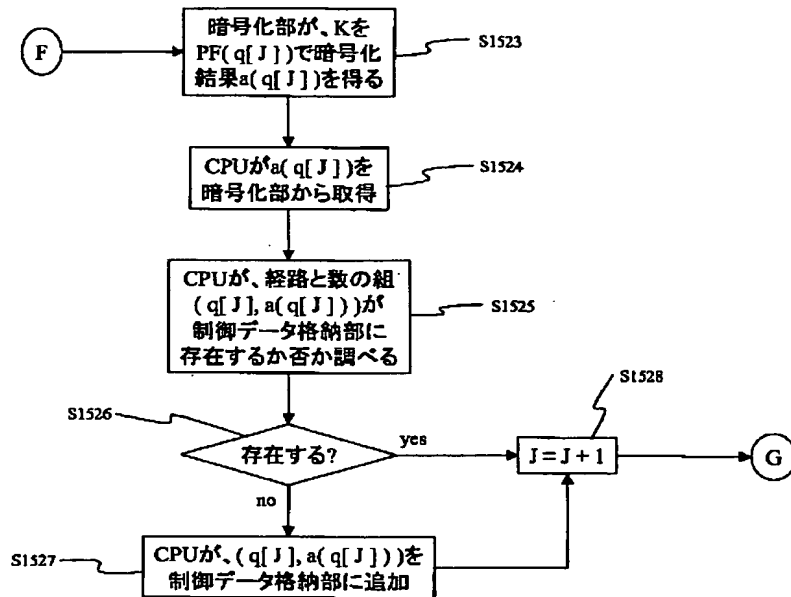
【図14】



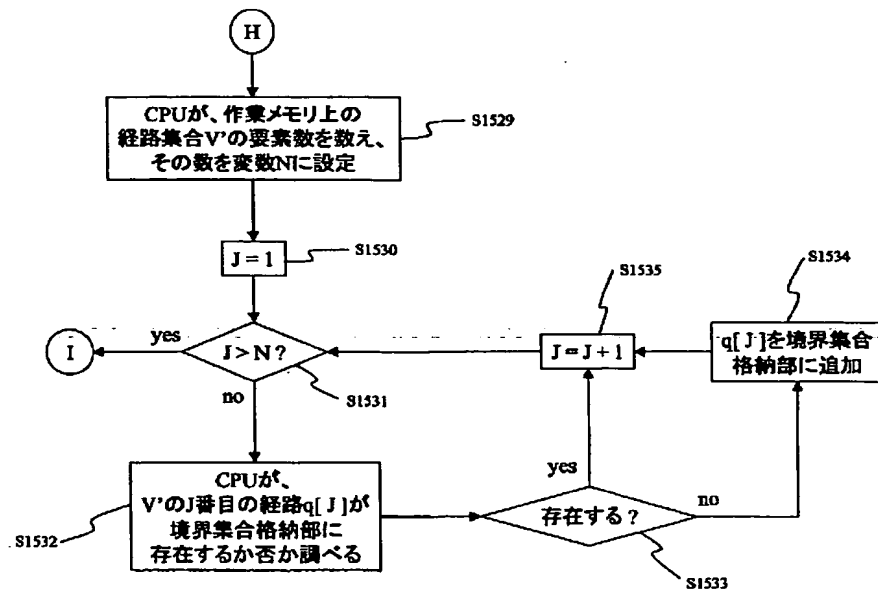
【図15】



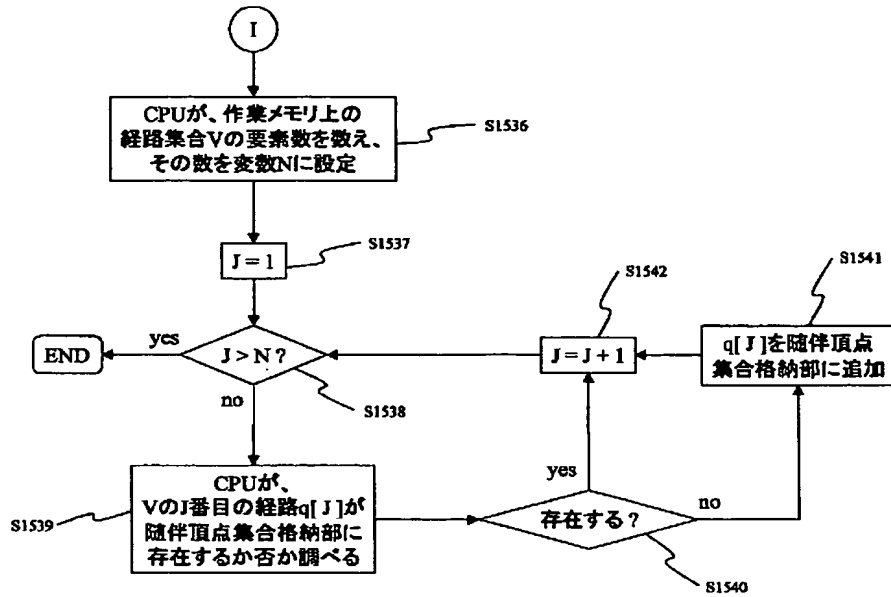
【図16】



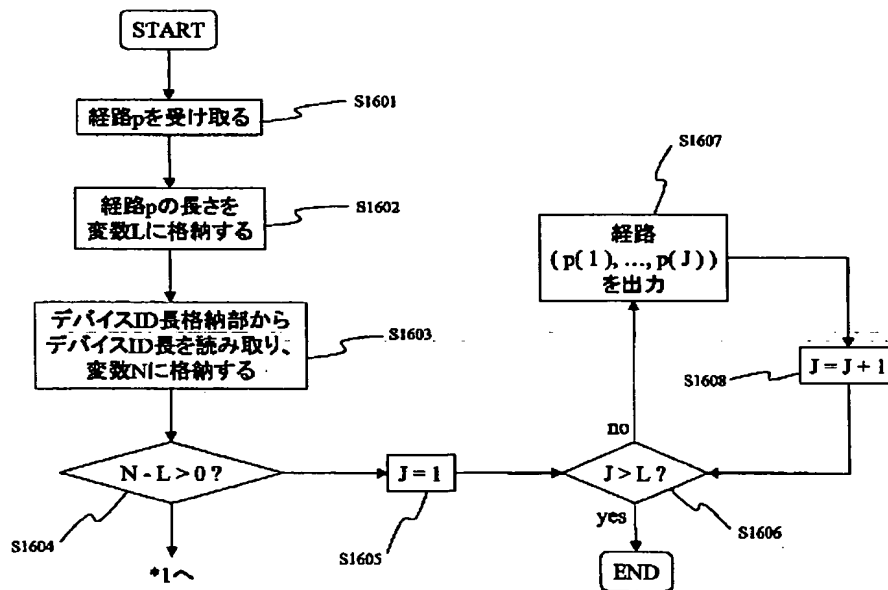
【図17】



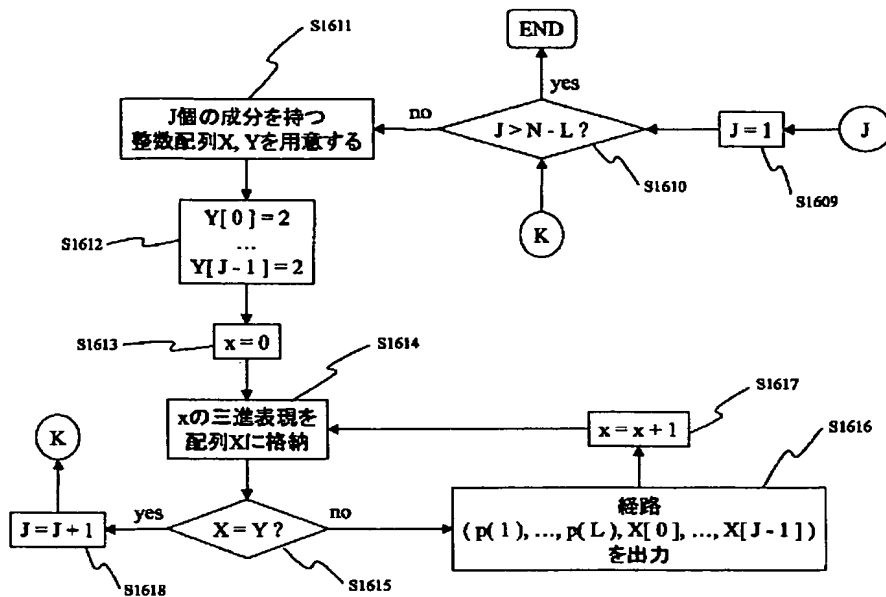
【図18】



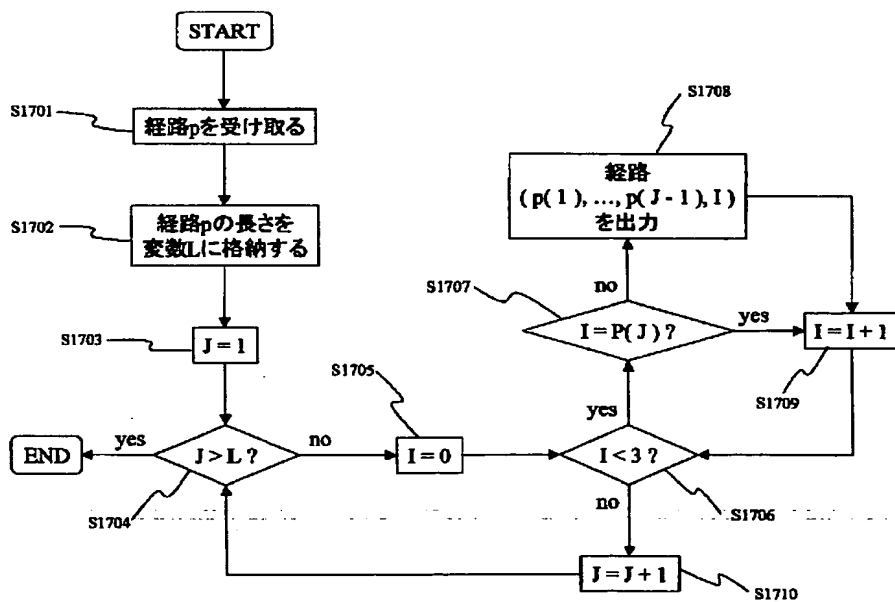
【図19】



【図20】



【図21】



【図39】

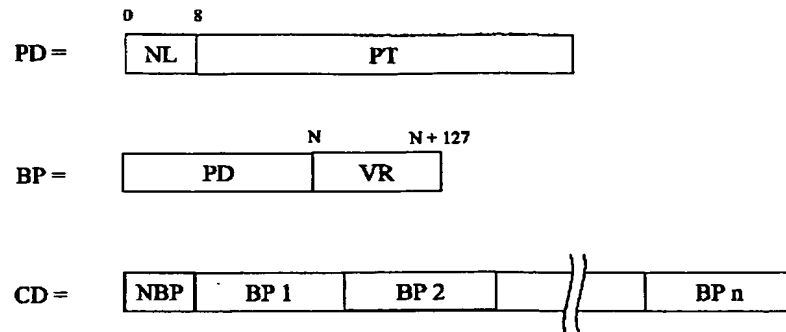
```

int      P[n];
NNum     KDP[n];

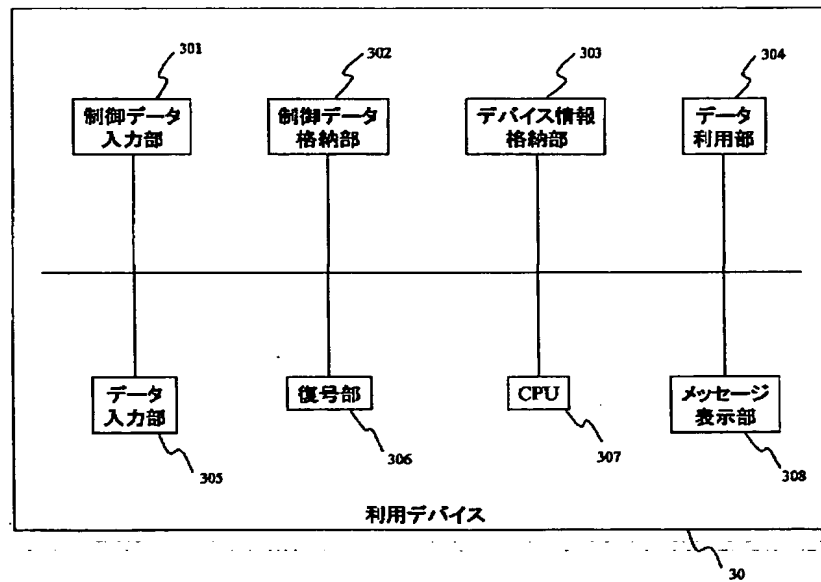
int      j;
NNum     Result;

for (j = 0; j < n; j++)
{
    Result = Dec(KDP[p[j], j], M[p[j], j]);
    if (Result != null)
        break;
}
  
```

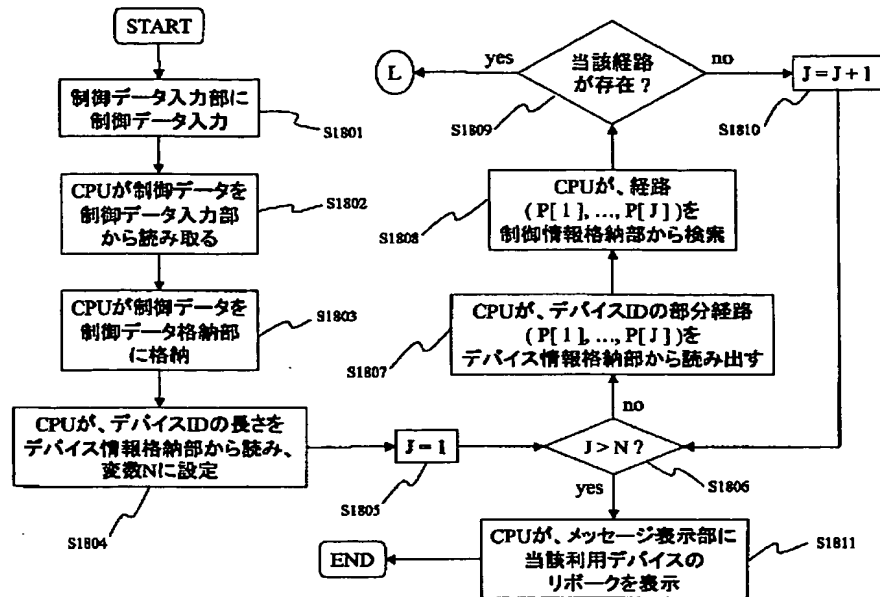
【図 2 2】



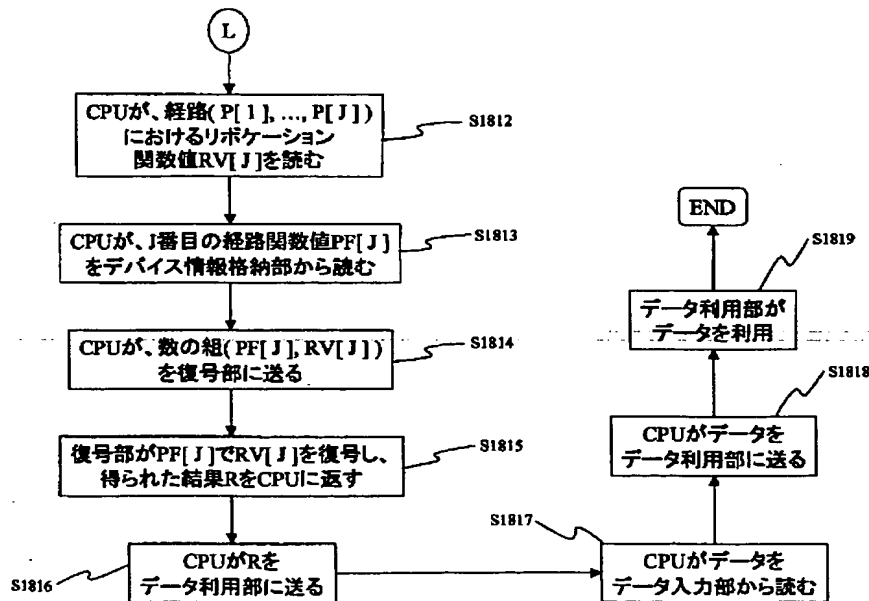
【図 2 3】



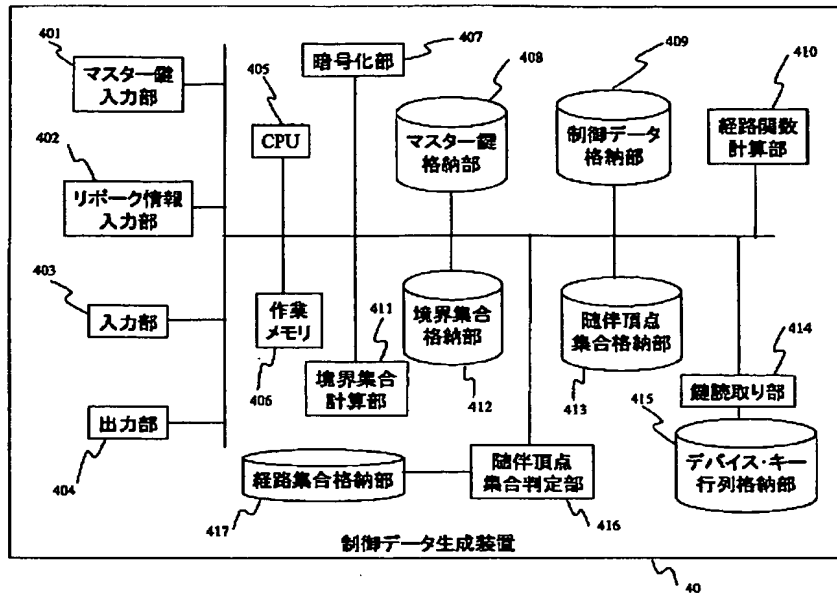
【図24】



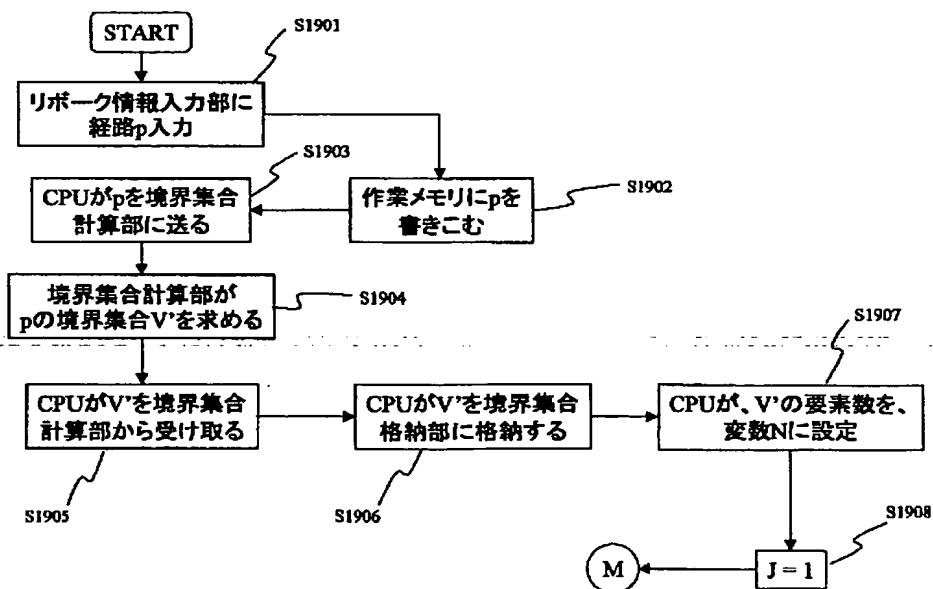
【図25】



【図26】

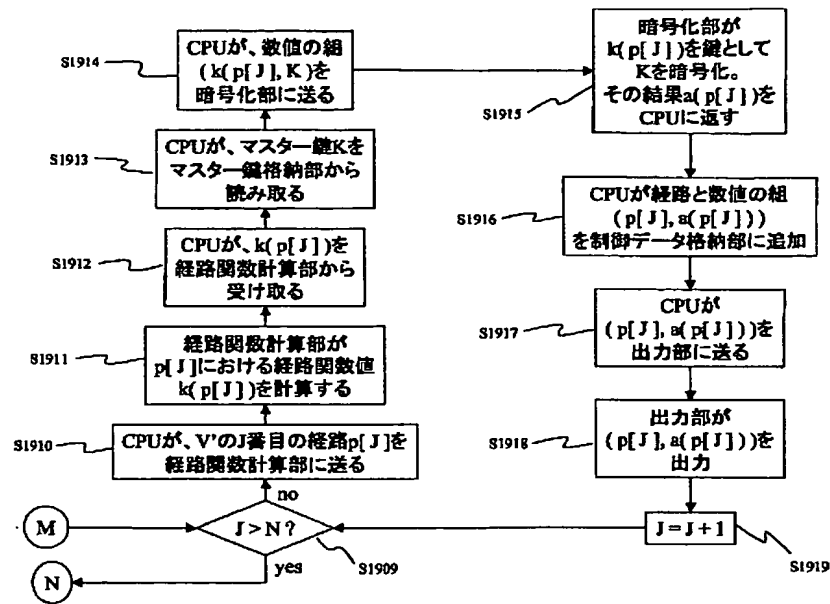


【図27】

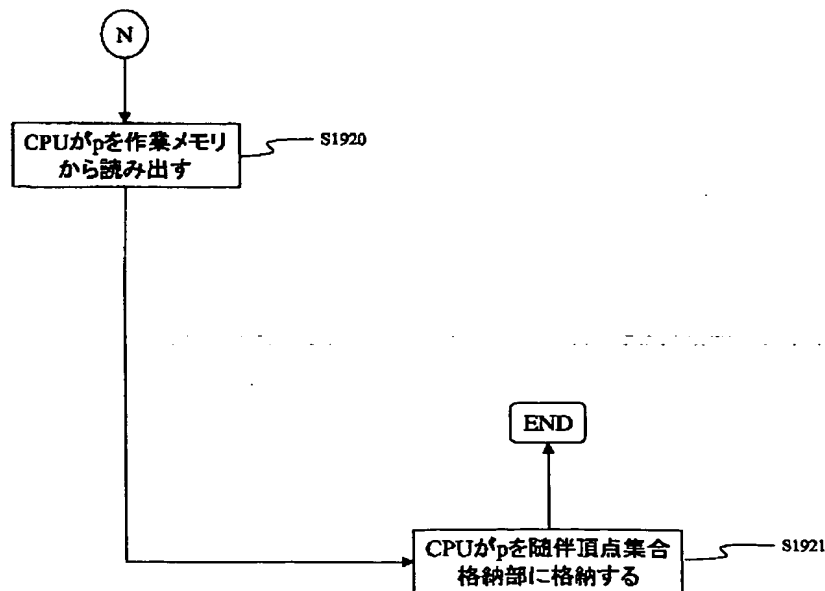




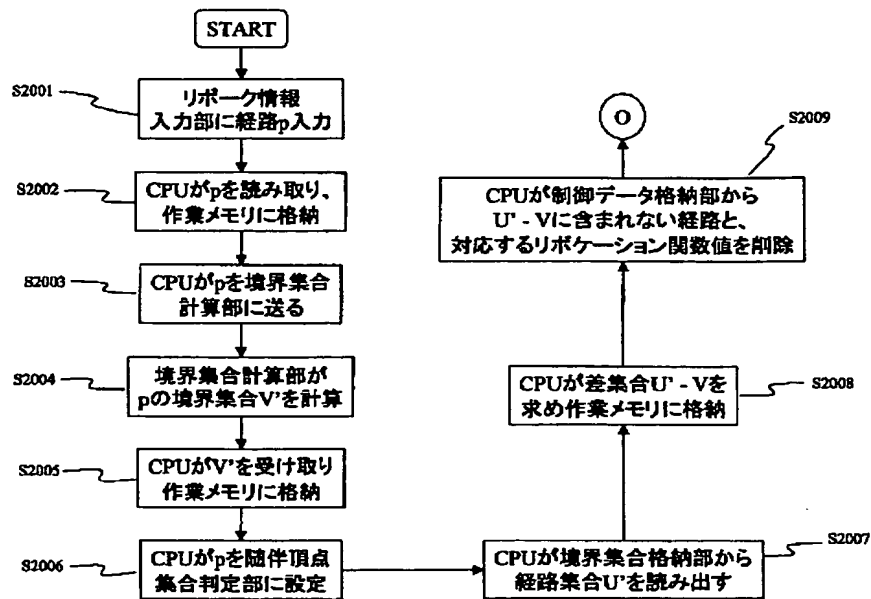
【図28】



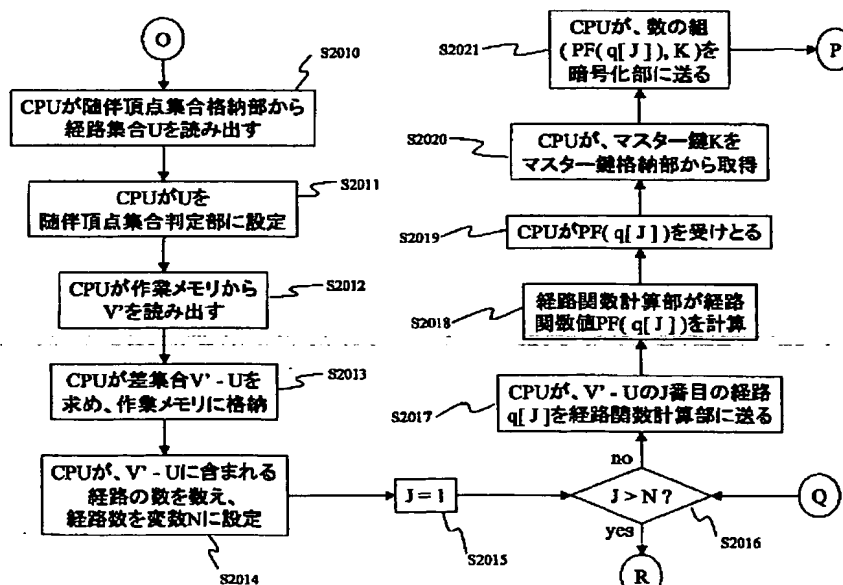
【図29】



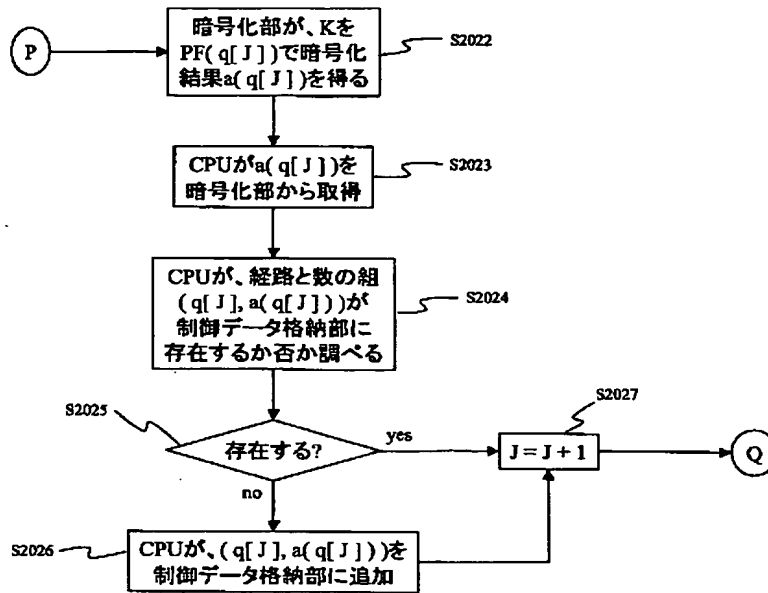
【図30】



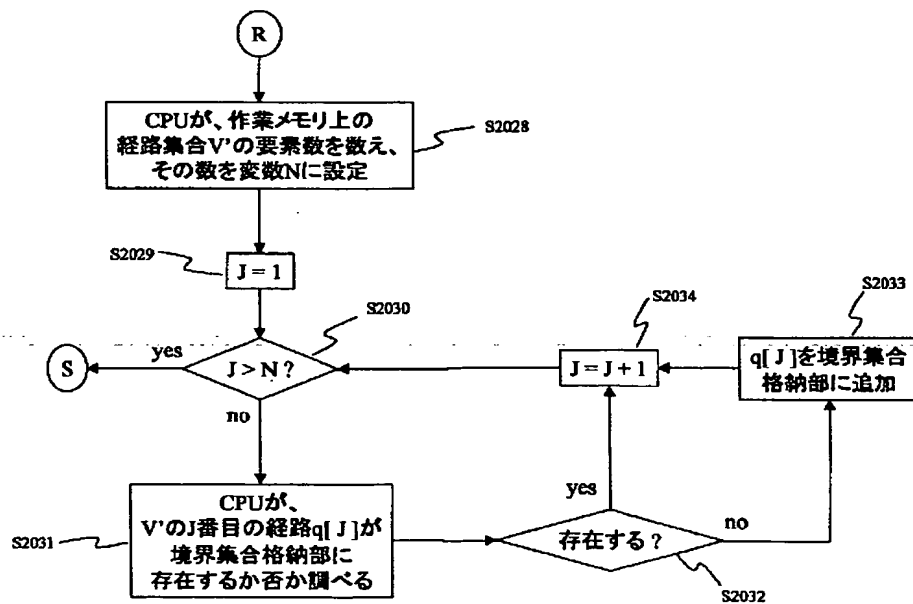
【図31】



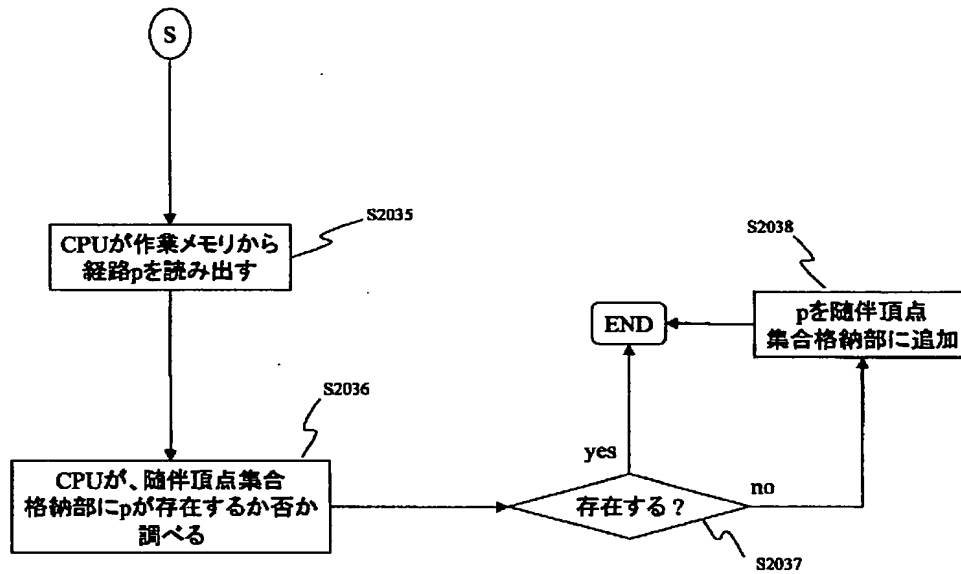
【図32】



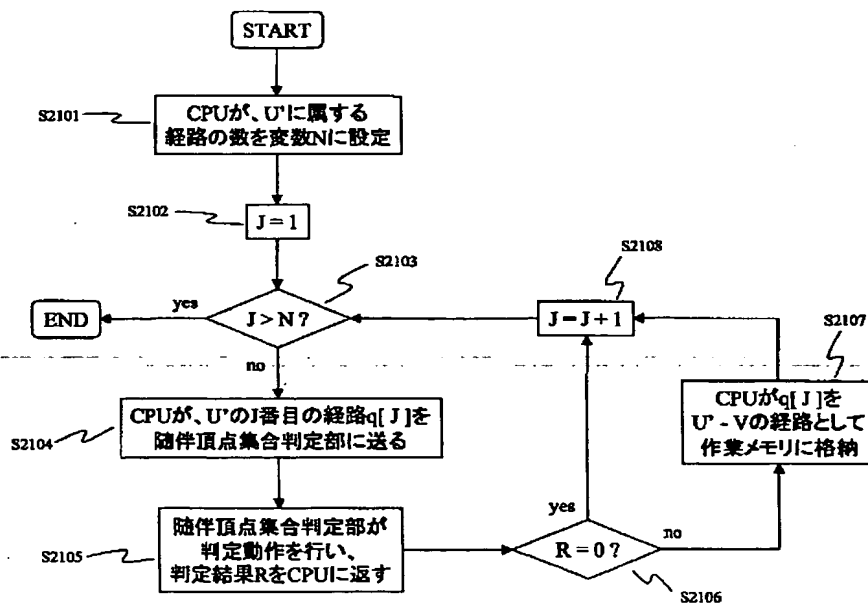
【図33】



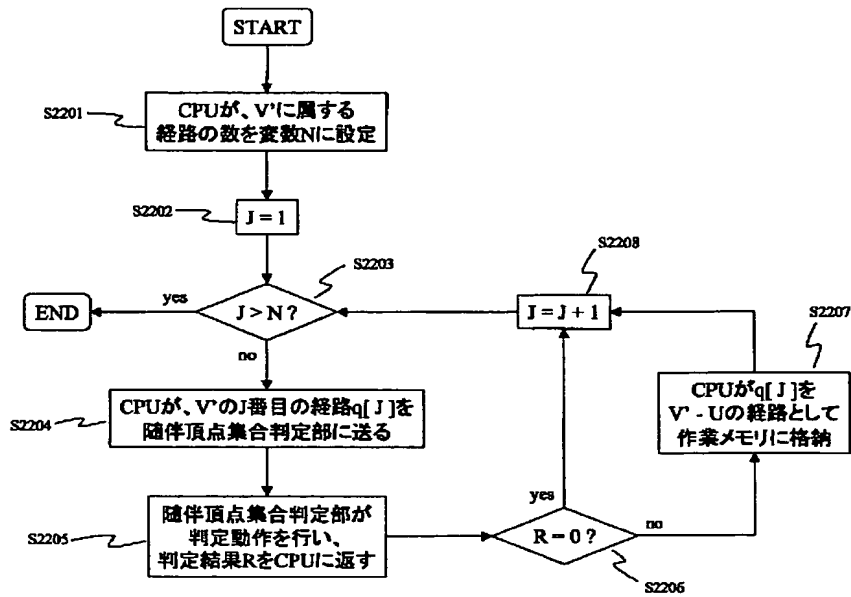
【図34】



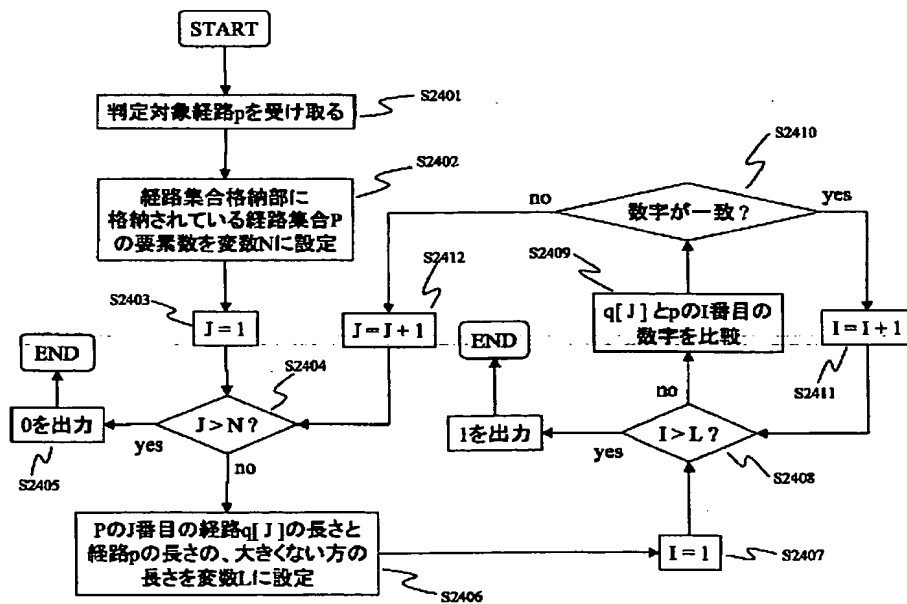
【図35】



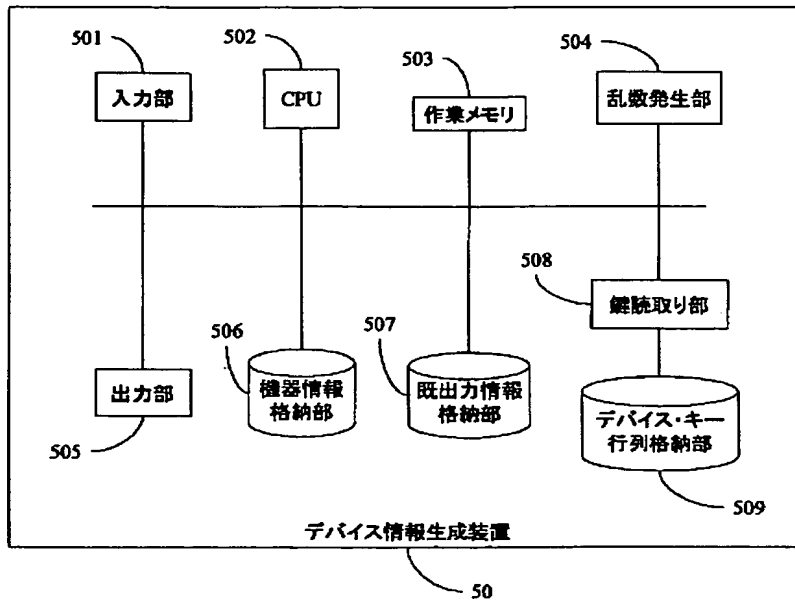
【図36】



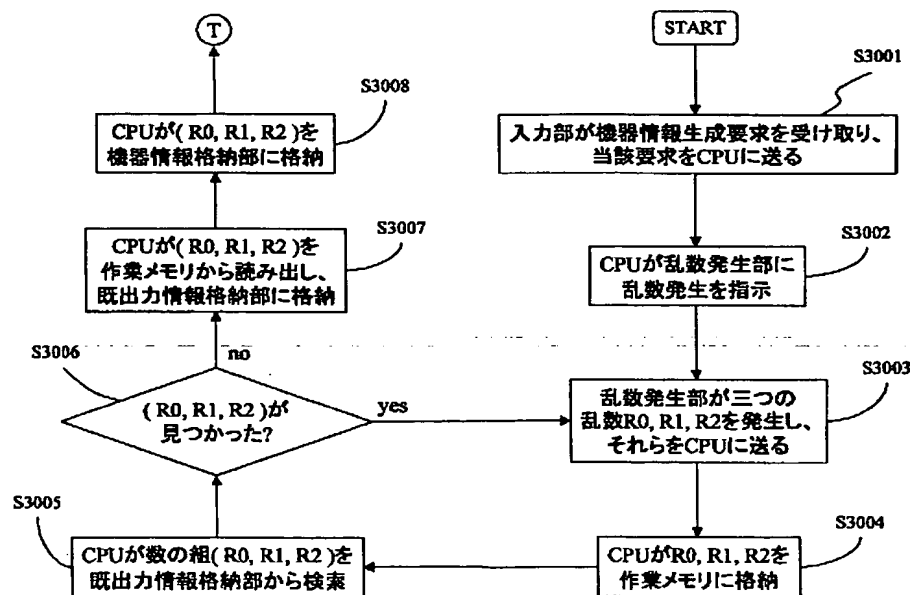
【図38】



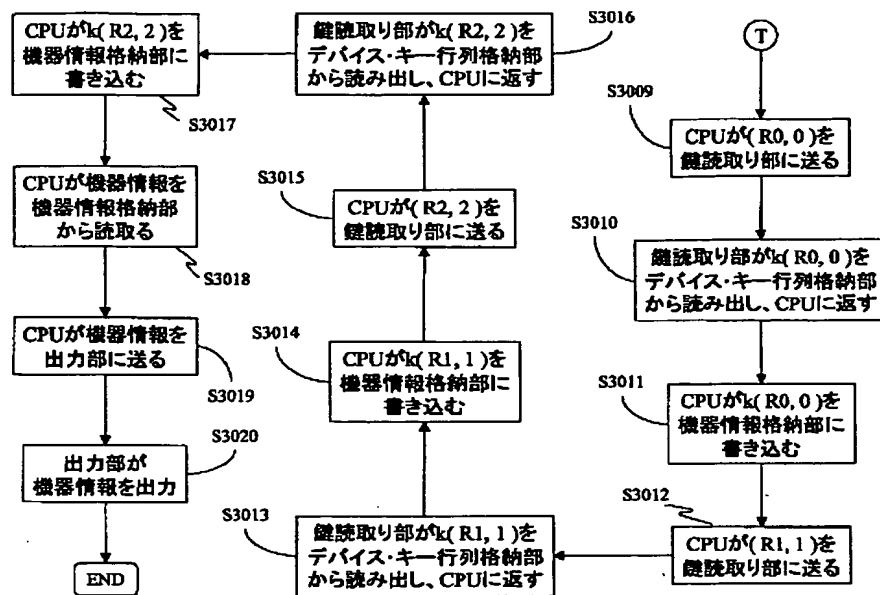
【図40】



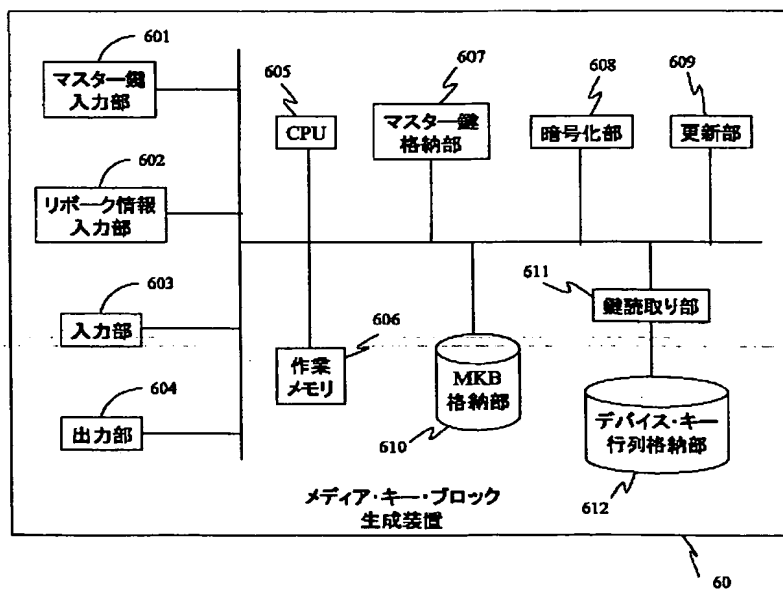
【図41】



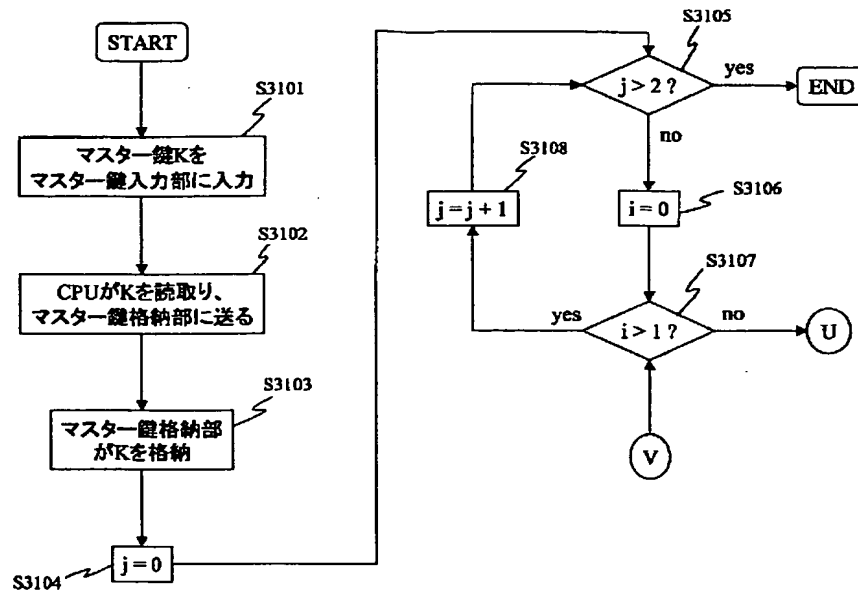
【図42】



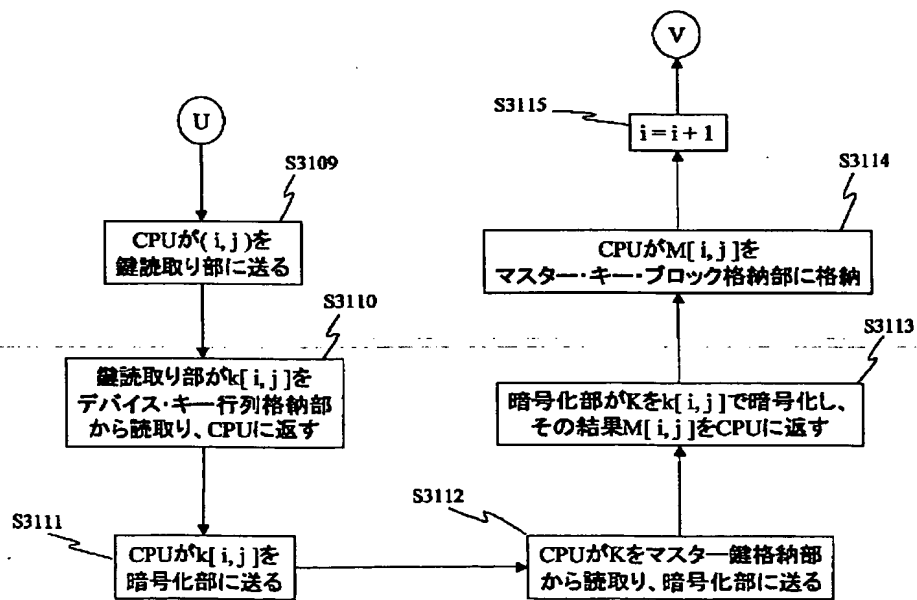
【図43】



【図44】

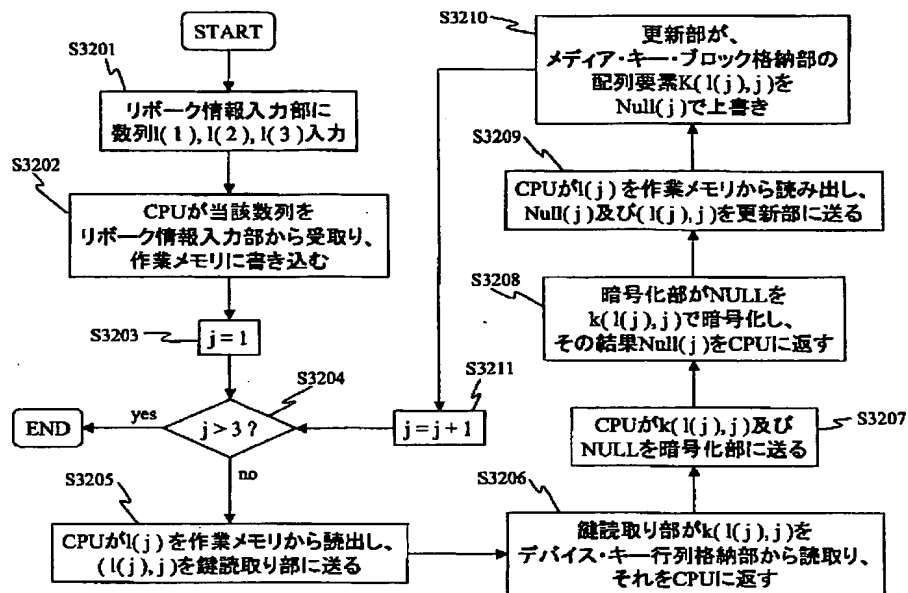


【図45】

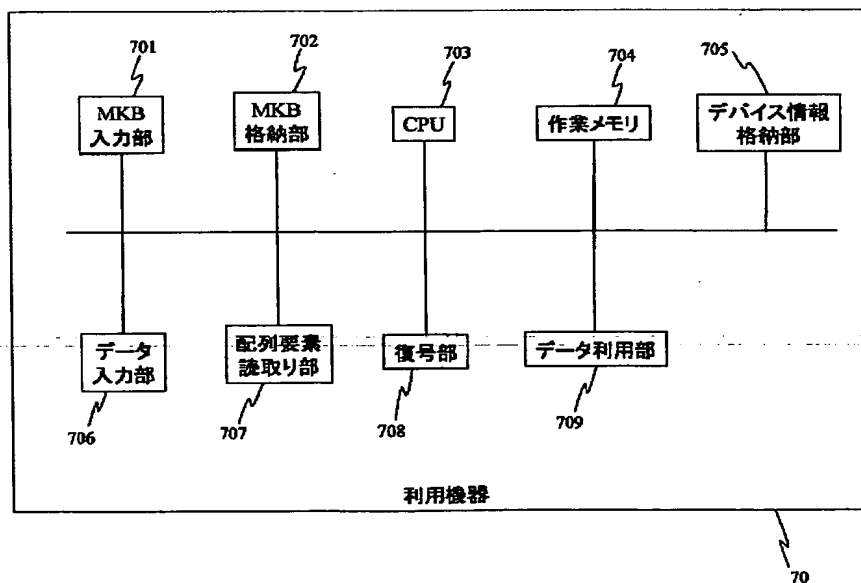




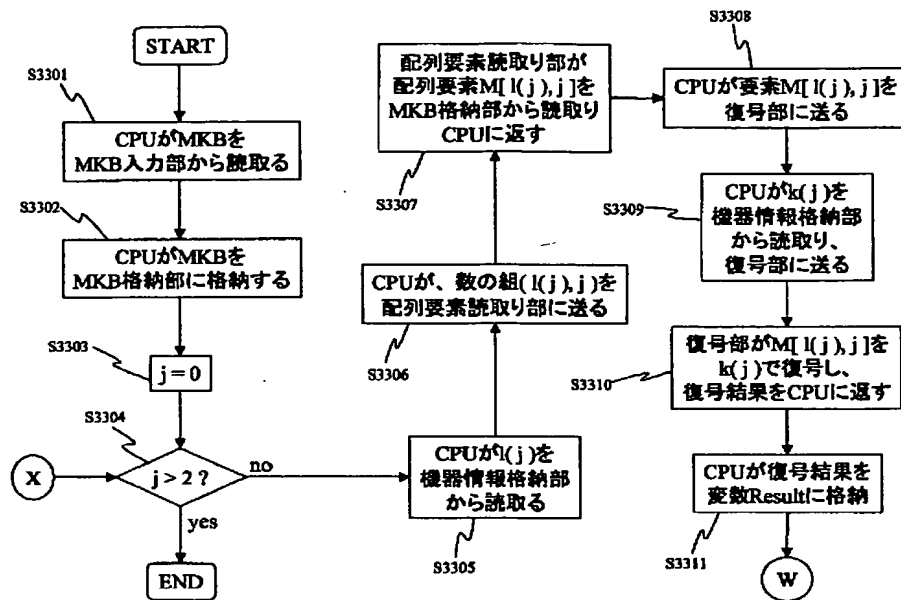
【図46】



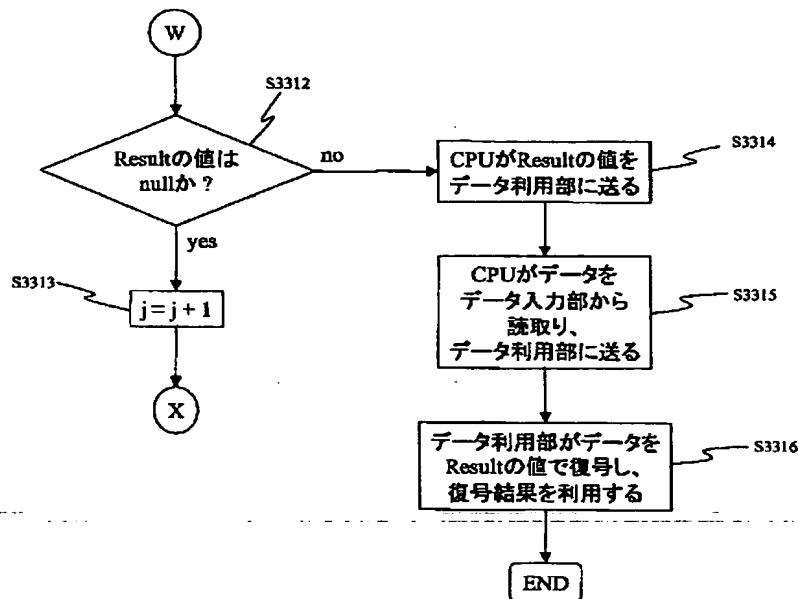
【図47】



【図48】



【図49】



**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☒ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**